TECHNICKÁ UNIVERZITA V LIBERCI
FAKULTA PŘÍRODOVĚDNĚ-HUMANITNÍ
A PEDAGOGICKÁ

Katedra: NTI
Studijní program: Učitelství všeobecně vzdělávacích předmětů pro SŠ
Studijní obor: Matematika – Informatika

VIZUALIZACE KONEČNÝCH AUTOMATŮ PRO VÝUKOVÉ ÚČELY

THE VISUALISATION OF FINITE AUTOMATA FOR EDUCATIONAL POURPOSES

Diplomová práce: *11-FP-NTI-002*

Autorka: **Podpis:**
Mgr. Kateřina Eichlerová

Vedoucí práce: Ing. Lenka Kosková - Třísková
Konzultant:

Počet

stran	grafů	obrázků	tabulek	pramenů	příloh
87	0	31	1	22	2

V Liberci dne: 29. 4. 2011

Zadání DP

ČESTNÉ PROHLÁŠENÍ

Název práce:	Vizualizace konečných automatů pro výukové účely
Jméno a příjmení autora:	Mgr. Kateřina Eichlerová
Osobní číslo:	P 08 000 999

Byla jsem seznámena s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména § 60 – školní dílo.

Prohlašuji, že má diplomová práce je ve smyslu autorského zákona výhradně mým autorským dílem.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracovala samostatně s použitím uvedené literatury a na základě konzultací s vedoucí diplomové práce.

Prohlašuji, že jsem do informačního systému STAG vložila elektronickou verzi mé diplomové práce, která je identická s tištěnou verzí předkládanou k obhajobě a uvedla jsem všechny systémem požadované informace pravdivě.

V Liberci dne: 29. 4. 2011

Kateřina Eichlerová

PODĚKOVÁNÍ

Velký dík patří mé rodině za opravdu silnou podporu při celém studiu a za to, že mi umožnili intenzivně se věnovat psaní této práce.

ANOTACE

Autorka v první části shrnuje základní pojmy z oblasti konečných deterministických automatů, bez nichž by čtenáři mohly být další partie nesrozumitelné. Druhá část je věnována některým již existujícím aplikacím, které mohou studentům při studiu konečných automatů pomoci. Samostatná kapitola je věnována možnosti popisu konečných deterministických automatů. Výklad je doplněn konkrétním příkladem. V poslední části je popsána webová aplikace, která byla vytvořena v rámci řešení této diplomové práce.

Klíčová slova: *deterministický konečný automat, stav, vstupní abeceda, slovo, přechodová funkce, stavový diagram, animace.*

ANNOTATION

In the first part, the autor summarizes basic ideas of the areas of deterministic finite automata, without which the reader could be a lot more incomprehensible. The second part is dedicated to some of already existing applications, which could help students in the study of deterministic finite automata. Separate chapter is about options of describing deterministic finite automata. Interpretation is supplemented by specific example. In the last part, the web application is described, which was created within the solution of this work.

Key words: *deterministic finite automaton, state, entry alphabet, word, transition function, state diagram, animation.*

OBSAH

SEZNAM POUŽITÝCH ZKRATEK A SYMBOLŮ

-	mínus
()	závorky
$\sum^+$	množina všech neprázdných slov
$\sum^*$	množina všech slov
{}	množinové závorky
|w|	délka slova
+	plus
=	je rovno
≠	není rovno
∃	existuje
∈	je prvkem
∑	vstupní abeceda, množina znaků
⊆	je podmnožinou
DKA	deterministický konečný automat
F	množina rozpoznávacích stavů
FSMXML	finite state machine XML
KA	konečný automat
Q	množina stavů
q_0	počáteční stav
SCXML	state chart XML
SVG	scaleable vector graphics
w	slovo
XML	extensible markup language
δ	přechodová funkce
ε	prázdné slovo

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

1. TEORETICKÝ ÚVOD

Konečný deterministický automat nad abecedou Σ je uspořádaná pětice: $\boldsymbol{KA = (Q, \Sigma, q_0, F, \delta)}$, kde Q je neprázdná množina stavů, Σ je neprázdná množina znaků – vstupní abeceda, $q_0 \in Q$ je počáteční stav, $F \subseteq Q$ je neprázdná množina rozpoznávacích stavů a δ je přechodová funkce, která jednoznačně přiřazuje stavu z množiny stavů a znaku ze vstupní abecedy stav z množiny stavů; $\delta: Q \times \Sigma \to Q$. Každý ze stavů má pro každý znak jednoznačně definovaný přechod.

Množinu všech slov nad abecedou Σ označujeme jako $\boldsymbol{\Sigma^*}$, **množinu všech neprázdných slov** $\boldsymbol{\Sigma^+}$, **prázdné slovo**, tedy slovo s nulovou délkou, $\boldsymbol{\varepsilon}$. Platí tedy, že $\boldsymbol{\Sigma^+ = \Sigma^* - \{\varepsilon\}}$. Prázdné slovo má délku 0, **délka** neprázdného **slova** $\boldsymbol{w}$ je počet jeho písmen a značí se $\boldsymbol{|w|}$.

Jazyk $\boldsymbol{L}$ **nad abecedou Σ** je libovolná množina slov, která jsou složena ze symbolů (písmen, znaků) vstupní abecedy Σ. Slova, která deterministický konečný automat rozpoznává (přijímá, akceptuje), bývají určena nějakou další podmínkou – např. čísla, která jsou dělitelná čtyřmi, slova, která končí na *abc*.

Deterministický automat postupně načítá symboly vstupní abecedy (slovo) a mění svůj stav podle definované přechodové funkce. Automat čte postupně jednotlivé znaky zleva. Pokud po načtení posledního znaku přejde do rozpoznávacího stavu, říkáme, že automat rozpozná (načte, akceptuje) slovo. **Deterministický automat rozpozná slovo** $\boldsymbol{w \in \Sigma}$, **jestliže existuje rozpoznávací stav** $\boldsymbol{q_r \in F;\ \delta(q_0, w) = q_r}$.

V množině stavů někdy existují stavy, jichž nelze dosáhnout libovolným přechodem z počátečního stavu. Tento **stav** je nazýván **nedosažitelný**. Symbolicky zapsáno: $\exists q \in Q; pro\ \forall w \in \Sigma, \delta(q_0, w) \neq q$. Tyto stavy nemají žádný vliv na přijímání jazyka, proto je můžeme ze zadání

eliminovat. Získáme tak jednodušší **automat**, který je s původním **ekvivalentní**, to znamená, že oba automaty rozpoznávají stejný jazyk.

Pro zjednodušení můžeme z návrhu automatu vyloučit i **stavy navzájem ekvivalentní**, které se při přijímání určitého slova chovají stejně. $\exists p, q \in Q; pro\ \forall w \in \Sigma, \delta(p, w) = \delta(q, w)$

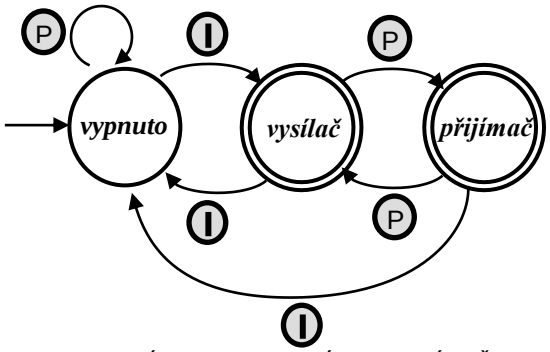
Odstranění ekvivalentních a nedosažitelných stavů se říká **redukce automatu**.

Příkladem konečného automatu může být *vysílačka*, která je buď *vypnutá*, zapnutá jako *vysílač*, nebo zapnutá ve funkci *přijímače*. Vstupy jsou pro vysílačku dvě tlačítka, která ji ovládají.

Na začátku je vysílačka vypnutá (stav *vypnuto*). Stiskem tlačítka Ⓘ ji převádíme do stavu *vysílač*, stiskem tlačítka Ⓟ přechází ze stavu *vysílač* do stavu *přijímač* nebo naopak.

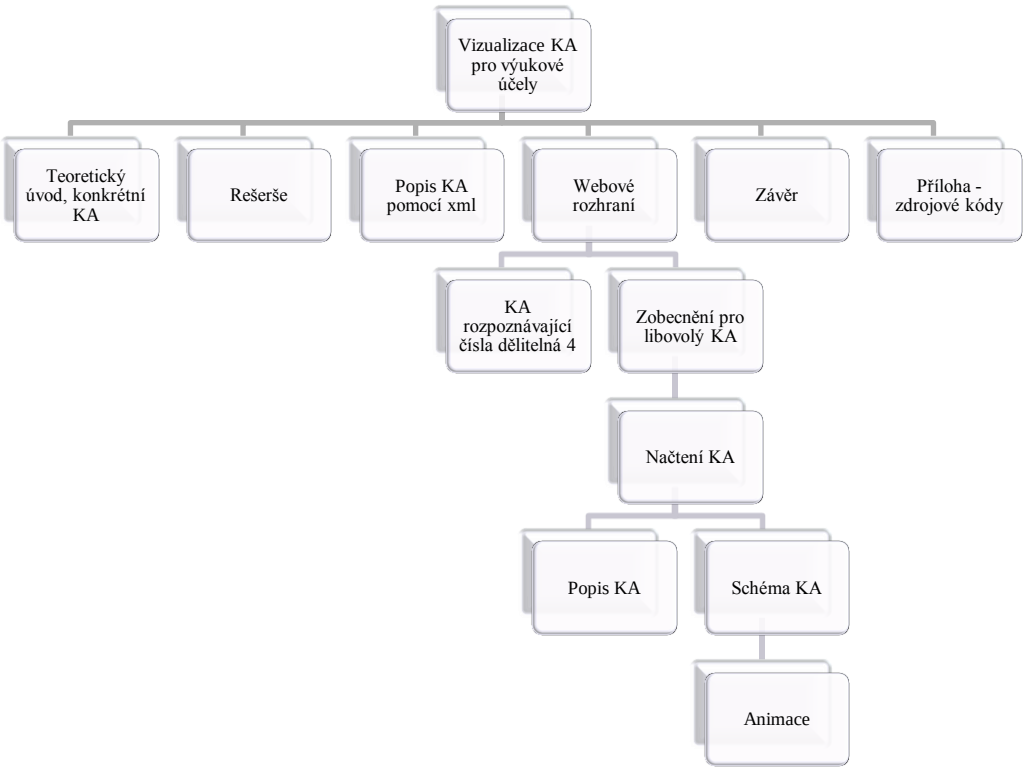
Obrázek 2 znázorňuje schéma, jak vysílačka reaguje na vstupy změnou stavu. Počáteční stav *vypnuto* je označen šipkou. Dvojitou kružnicí jsou zvýrazněny rozpoznávací stavy, tedy stavy, které jsou pro uživatele nějak důležité – pokud se chci s někým vysílačkou domluvit, musí být buď ve funkci *vysílače*, nebo *přijímače*.

Počáteční stav je právě jeden, rozpoznávacích stavů může být i více, vždy alespoň jeden.



OBRÁZEK 1 – SCHÉMA VYSÍLAČKY

SCHÉMA DIPLOMOVÉ PRÁCE



OBRÁZEK 2 – SCHÉMA DIPLOMOVÉ PRÁCE

2. MOŽNOSTI POPISU KONEČNÉHO AUTOMATU

2.1. POPIS LIBOVOLNÉHO KONEČNÉHO AUTOMATU

Jak víme, deterministický konečný automat je uspořádaná pětice $KA = (Q, \Sigma, q_0, F, \delta)$. Pokud chceme popsat nějaký automat, musíme definovat všech pět jeho částí, tedy množinu stavů, vstupní abecedu, počáteční stav, množinu rozpoznávacích stavů a přechodovou funkci. Přechodovou funkci můžeme popsat několika způsoby (v kapitole 2.2 jsou všechny způsoby demonstrovány na konkrétním příkladu):

- **Výpisem**
- **Tabulkou**
- **Stavovým diagramem**
- **Stavovým stromem**

Celý automat můžeme pro počítačové zpracování popsat:

- **S pomocí jazyka XML**

Pro popis konečných automatů byl konsorciem W3C vyvinut jazyk SCXML [2], který autorka přizpůsobila potřebám práce. Podobně jako SCXML je navržen i jazyk FSMXML [16]. Prvkem, který zabránil využití již definovaných jazyků, byl atribut *final* elementu *state* (koncový stav), který nepřipouští další přechod ze stavu, jenž má tuto vlastnost přiřazenu. Pokud bychom takový atribut povolili, nemohli bychom například v konečném automatu rozpoznávajícím čísla dělitelná čtyřmi zadat žádné číslo, které obsahuje dvojčíslí *lichá číslice + 0, 4*, nebo *8*, ani žádné dvojčíslí *sudá číslice + 2*, nebo *6*. V takovém případě by automat přešel do koncového stavu již po prvním načtení dané kombinace znaků a dál by žádnou číslici nenačetl.

Popis dialektu pomocí Relax NG [13]:

```
<element xmlns="http://relax.org/ns/structure/1.0" name="FA">
  <attribute name="name"/>
  <attribute name="startstate"/>
  <oneOrMore>
    <element name="state">
      <attribute name="id"/>
      <attribute name="accepting"/>
      <oneOrMore>
        <element name="transition">
          <attribute name="symbol"/>
          <attribute name="target"/>
        </element>
      </oneOrMore>
    </element>
  </oneOrMore>
</element>
```

Kořenový element ***FA*** obsahuje právě jeden atribut – počáteční stav (***startstate***) a jeden nebo více elementů stav (***state***). Každý stav obsahuje atributy ***id*** a atribut ***accepting***, který určuje, zda je stav rozpoznávací. Pro každý stav je dále definován element přechod ***(transition***), který svými atributy určuje, pro jaký znak (***symbol***) je určen a jaký je stav po přechodu (***target***).

2.2. DETERMINISTICKÝ KONEČNÝ AUTOMAT ROZPOZNÁVAJÍCÍ ČÍSLA DĚLITELNÁ ČTYŘMI

Čísla, která jsou dělitelná čtyřmi, se poznají podle posledního dvojčíslí; pokud je poslední dvojčíslí dělitelné čtyřmi, je i celé číslo dělitelné čtyřmi. Na místě desítek může být buď sudá, nebo lichá číslice. Pokud je číslice na místě desítek sudá a celé číslo má být dělitelné čtyřmi, musí být na místě jednotek 0, 4, nebo 8. V případě, že je na místě desítek číslice lichá, číslo musí končit na 2, nebo 6.

Číslice jsou tedy rozděleny do tří skupin: $\boldsymbol{L} = \{\mathbf{1}, \mathbf{3}, \mathbf{5}, \mathbf{7}, \mathbf{9}\}$, $\boldsymbol{S} = \{\mathbf{2}, \mathbf{6}\}$ a $\boldsymbol{D} = \{\mathbf{0}, \mathbf{4}, \mathbf{8}\}$. Znaky L, S, D tvoří vstupní abecedu. Množina stavů je $Q = \{licha, suda, deli\}$, kde stav $\boldsymbol{licha}$ reprezentuje čísla lichá, stav $\boldsymbol{suda}$ zastupuje sudá čísla, která nejsou dělitelná čtyřmi a stav $\boldsymbol{deli}$ je vyhrazen pro čísla, která čtyřmi dělitelná jsou.

Tento konkrétní konečný automat popíšeme:

Množina stavů $\boldsymbol{Q} = \{licha, suda, deli\}$

Vstupní abeceda $\boldsymbol{\Sigma} = \{D, L, S\}$

Počáteční stav $\boldsymbol{q_0} = deli$

Množina rozpoznávacích stavů $\boldsymbol{F} = \{deli\}$

Přechodová funkce δ definovaná výpisem:

$\delta(deli, L) = licha$

$\delta(deli, S) = suda$

$\delta(deli, D) = deli$

$\delta(licha, L) = licha$

$\delta(licha, S) = deli$

$\delta(licha, D) = suda$

$\delta(suda, L) = licha$

$\delta(suda, S) = suda$

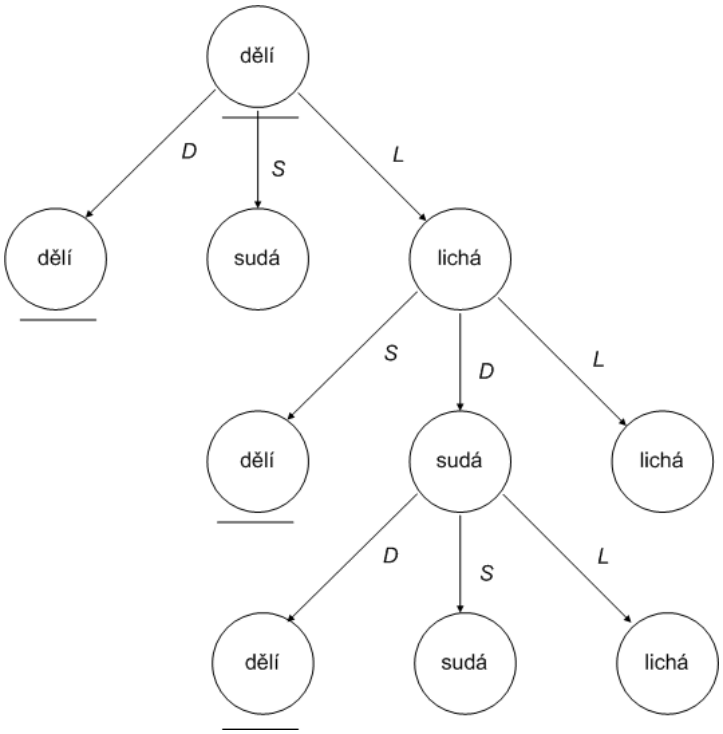
$\delta(suda, D) = deli$

Popis tabulkou (stav *deli* je počáteční i rozpoznávací):

	D	**L**	**S**
licha	suda	licha	deli
suda	deli	licha	suda
deli	deli	licha	suda

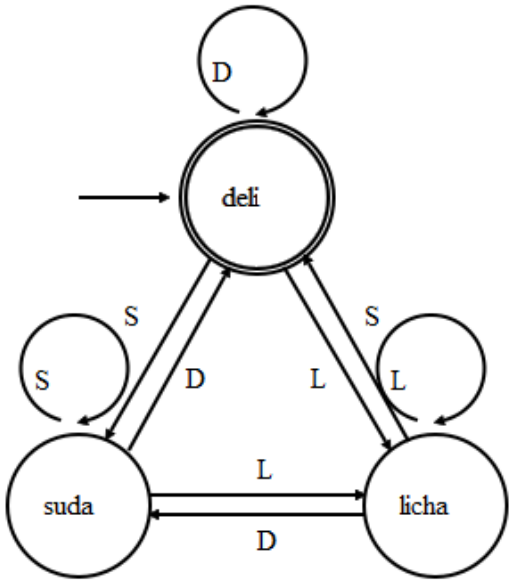
TABULKA 1 – POPIS PŘECHODOVÉ FUNKCE TABULKOU

Zobrazení stavovým stromem (počáteční stav je nejvýše, rozpoznávací stavy jsou podtrženy):



OBRÁZEK 3 – STAVOVÝ STROM

Znázornění stavovým diagramem (počáteční stav je označen šipkou ke stavu, rozpoznávací dvojitou kružnicí), kruhy zobrazují jednotlivé stavy, šipky přechody:



OBRÁZEK 4 – STAVOVÝ DIAGRAM

Popis automatu s pomocí navrženého dialektu jazyka XML:

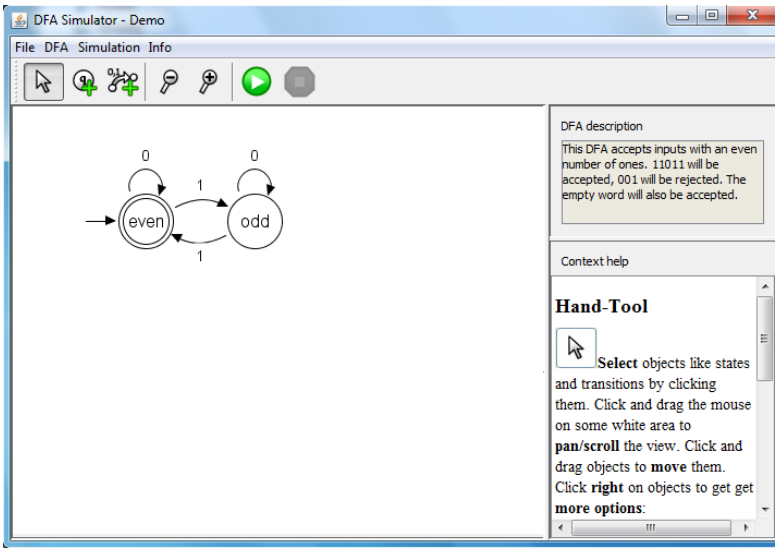
```
<FA name="delitelnost_4" startstate="deli">
    <state id="deli" accepting="true">
        <transition symbol="L" target="licha"/>
        <transition symbol="S" target="suda"/>
        <transition symbol="D" target="deli"/>
    </state>
    <state id="licha" accepting="false">
        <transition symbol="L" target="licha"/>
        <transition symbol="S" target="deli"/>
        <transition symbol="D" target="suda"/>
    </state>
    <state id="suda" accepting="false">
        <transition symbol="L" target="licha"/>
        <transition symbol="S" target="suda"/>
        <transition symbol="D" target="deli"/>
    </state>
</FA>
```

3. REŠERŠE

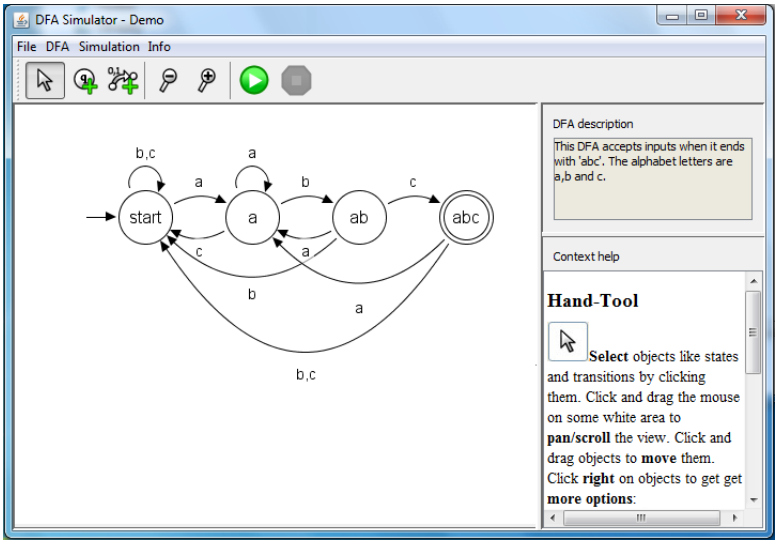
3.1. DFA_SIMULATOR.JAR

3.1.1. POPIS APLIKACE

DFA_simulator.jar [3] je Javová aplikace, která umožňuje uživateli znázornit libovolný stavový diagram přidáváním stavů (kruhů) a přechodů (šipek). Povolené znaky jsou číslice a písmena. V úvodní nabídce si uživatel může prohlédnout příklad automatu, který rozpoznává sudá čísla, nebo automatu, který rozpoznává slova končící na *abc*.

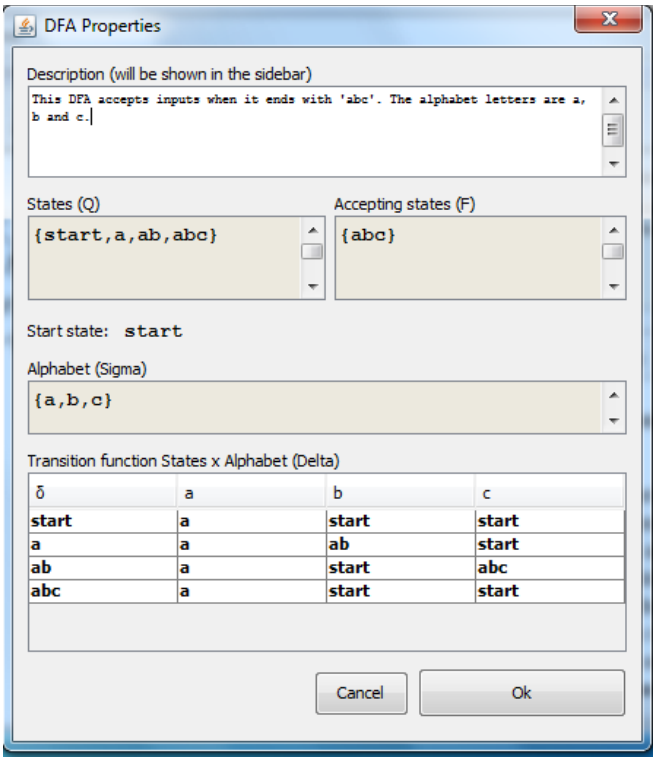


OBRÁZEK 5 – KA ROZPOZNÁVAJÍCÍ SUDÁ ČÍSLA



OBRÁZEK 6 – KA ROZPOZNÁVAJÍCÍ SLOVA, KTERÁ KONČÍ NA *abc*

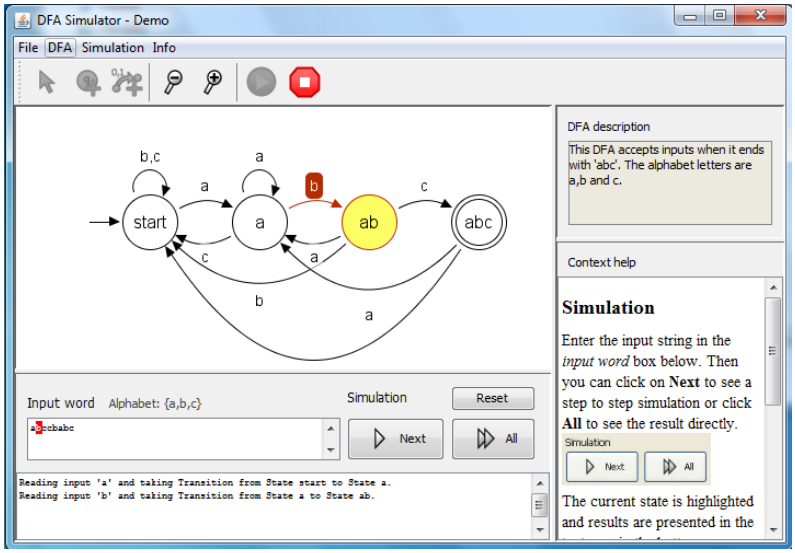
Chceme-li zobrazit informace o automatu, klepneme na nabídku *DFA*, kde vybereme příkaz *Descritpion and definition*… Objeví se nové okno, ve kterém jsou přehledným způsobem vypsány všechny prvky, které tvoří zadaný deterministický konečný automat a jeho stručný popis.



OBRÁZEK 7 – VLASTNOSTI KA

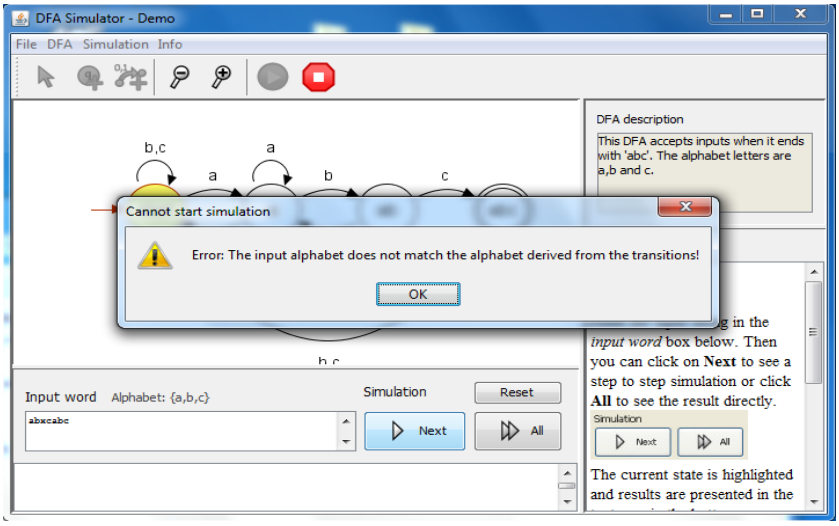
Nabídka *DFA* dále umožňuje automaticky doplnit neúplné zadání o stavy i znaky, minimalizovat automat a exportovat obrázek stavového diagramu ve formátu PNG.

Po klepnutí na zelenou šipku nebo na nabídku *Simulation – Start simulation* se zobrazí okno, ve kterém je možno zadat slovo (vstupní abeceda je přehledně popsána) a spustit animaci krok za krokem tlačítkem *Next*. Na modelu je vidět, jak KA postupně přechází z jednoho stavu do druhého. Barevně se postupně zvýrazní přechod a nový stav. Pokud je po načtení slova aktivní (zvýrazněný) rozpoznávací stav, objeví se informace o tom, že automat dané slovo přijímá. Tlačítko *All* zobrazí výpis a ihned se zvýrazní stav, na který automat přejde po načtení posledního znaku.



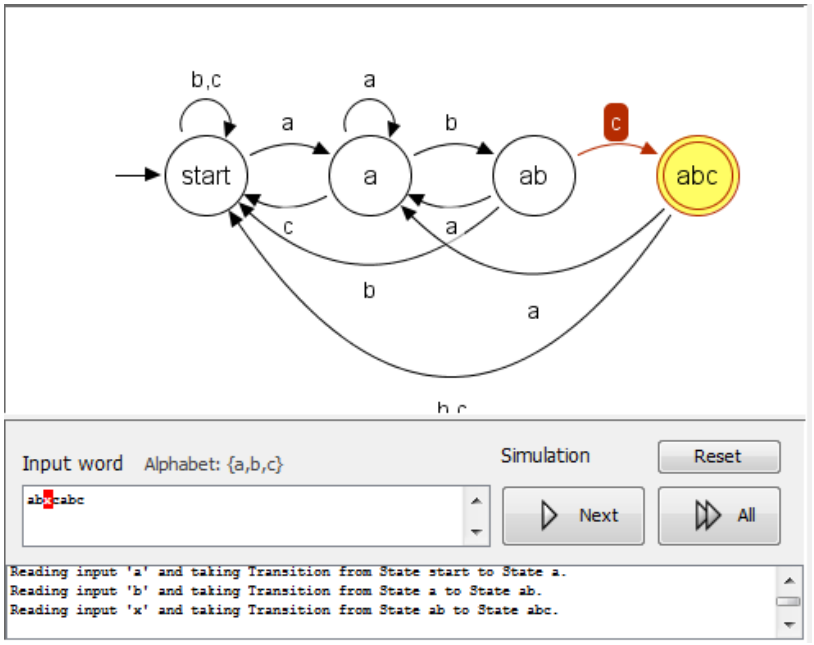
OBRÁZEK 8 – ANIMACE KA

V případě, že uživatel zadá znak, který není prvkem vstupní abecedy, objeví se upozornění na tento jev a animace se nespustí. Neplatný znak by zde mohl být zvýrazněn – písmo je poměrně malé a při obsáhlejší vstupní abecedě v kombinaci s delším zadávaným slovem by nemuselo být snadné chybu najít.



OBRÁZEK 9 – UPOZORNĚNÍ NA NEPLATNÝ ZNAK

Pokud ovšem uživatel smaže výpis a k již zapsanému slovo doplní další znaky, kontrola neproběhne a animace se spustí s původním slovem, i když výpis zobrazuje nové (i neplatné) znaky. V obou případech je zadáno slovo: *ab**x**cabc*.

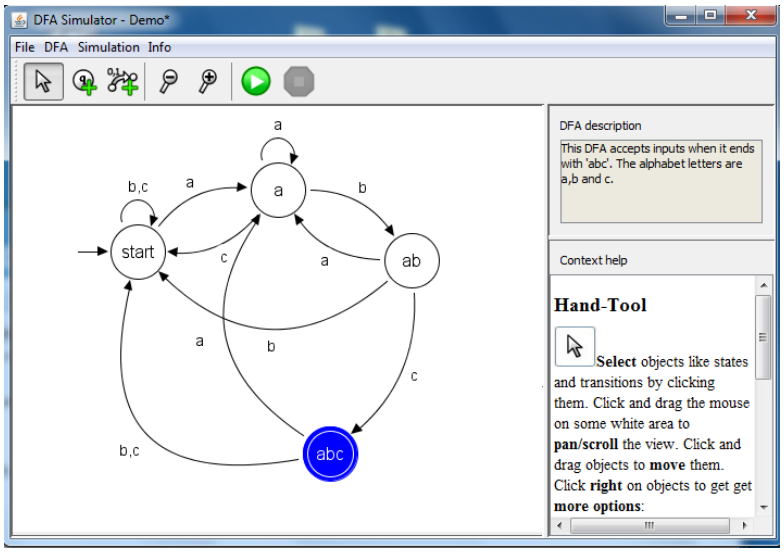


OBRÁZEK 10 – CHYBNÁ ANIMACE

Po klepnutí na červený osmiúhelník přepneme animaci do editačního módu, kde můžeme automat upravit. Můžeme přidat stavy, přechody, změnit umístění jednotlivých objektů tak, aby byl stavový diagram (ne)přehledný.

Klady

Mezi silné stránky programu patří možnost automatických oprav při zadávání stavů a přechodů vybraného deterministického konečného automatu, export stavového diagramu ve formě obrázku. Velmi příjemná je i možnost změny rozložení stavů pouhým tažením myší – šipky se automaticky posouvají s kruhem. Ne každé jiné uspořádání je však přehledné.



OBRÁZEK 11 – MOŽNOST ZMĚNY ROZLOŽENÍ STAVOVÉHO DIAGRAMU

Zápory

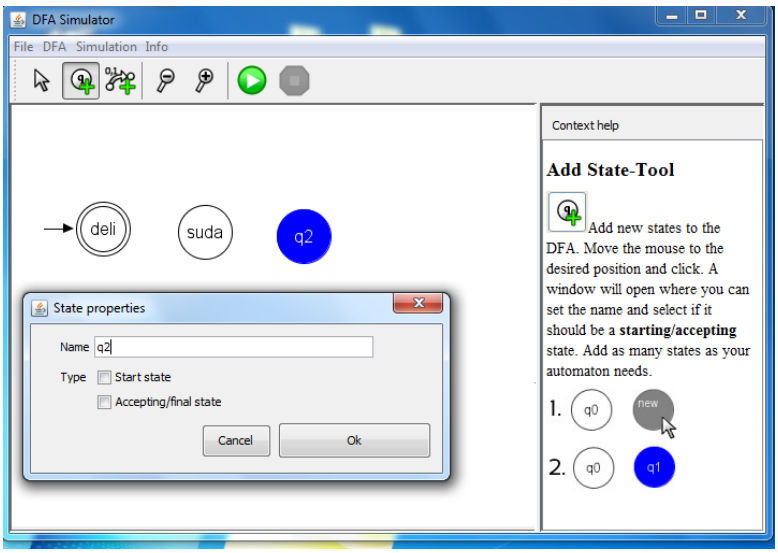
Autorka je toho názoru, že je didakticky vhodnější postupovat opačným směrem, tedy od popisu automatu k jeho grafickému znázornění, což tato aplikace neumožňuje. I v praxi, když je potřeba vytvořit návrh nějakého automatu, známe množinu stavů, vstupní abecedu, počáteční stav a množinu rozpoznávacích stavů. Přechodovou funkci definujeme podle zadání a zobrazení stavovým diagramem je „jen“ přehledné zobrazení návrhu.

Dalším negativem je již zmiňovaná možnost zadat slovo, které není složeno ze znaků vstupní abecedy a následné spuštění animace, která neodpovídá slovnímu popisu.

Špatně čitelné je velmi malé písmo některých popisků a v poli pro zadávání slova.

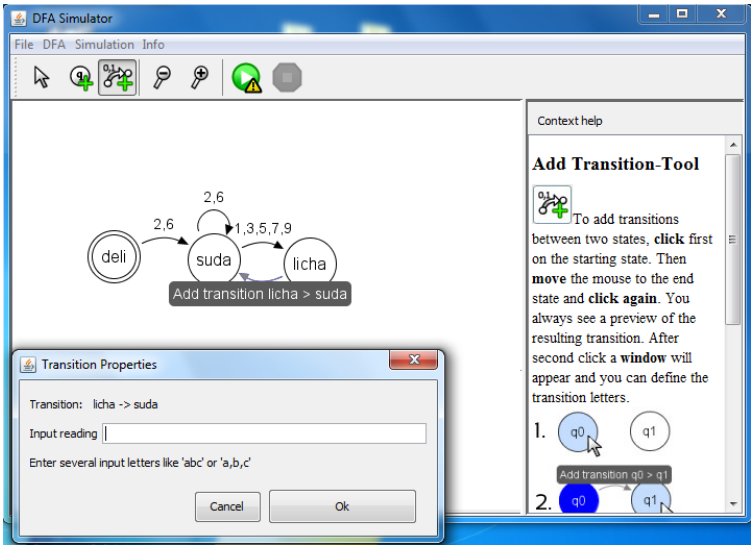
3.1.2. IMPLEMENTACE DKA ROZPOZNÁVAJÍCÍHO ČÍSLA DĚLITELNÁ 4

Pokud chceme v aplikaci vytvořit návrh vlastního DKA, musíme nejprve zobrazit a definovat jednotlivé stavy a přechody mezi nimi. Stavy se vkládají klepnutím do pracovní plochy po stisknutí tlačítka  . Pro každý stav se zobrazí dialogové okno, do kterého uživatel doplní název stavu a zda je stav počáteční nebo rozpoznávací. Program umožňuje označit více stavů za rozpoznávací, ale pouze jeden za počáteční. V případě, že není označen žádný stav za počáteční při zadávání diagramu, chybu nahlásí až při pokusu o spuštění animace. Animace nebude spuštěna, dokud uživatel počáteční stav nezadá (viz vykřičník u animace, který je vidět na dalším obrázku).



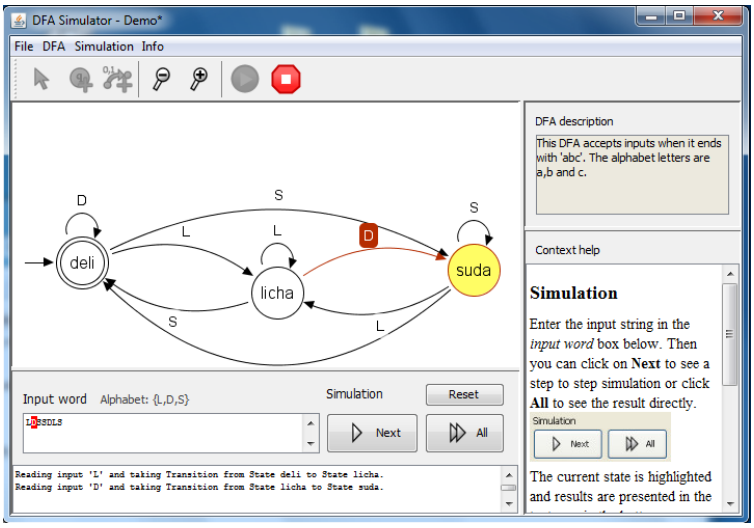
OBRÁZEK 12 – VYTVOŘENÍ NOVÉHO STAVU

Přechod mezi stavy zobrazíme po stisknutí tlačítka  , nejdřív klepneme na stav před přechodem a poté na stav po přechodu. Opět se zobrazí dialogové okno, do kterého vypíšeme jeden nebo více znaků, po jejichž stisknutí nastane zakreslovaný přechod.



OBRÁZEK 13 – VYTVOŘENÍ PŘECHODU

Zadat tento konečný automat je poměrně snadné. Obtížnější bylo vybrat rozložení stavového diagramu, aby bylo schéma přehledné. Pro zadání pomocí čísel to bylo prakticky nemožné, proto autorka zvolila již dříve uváděnou variantu rozdělení číslic do tří skupin D, S, L.

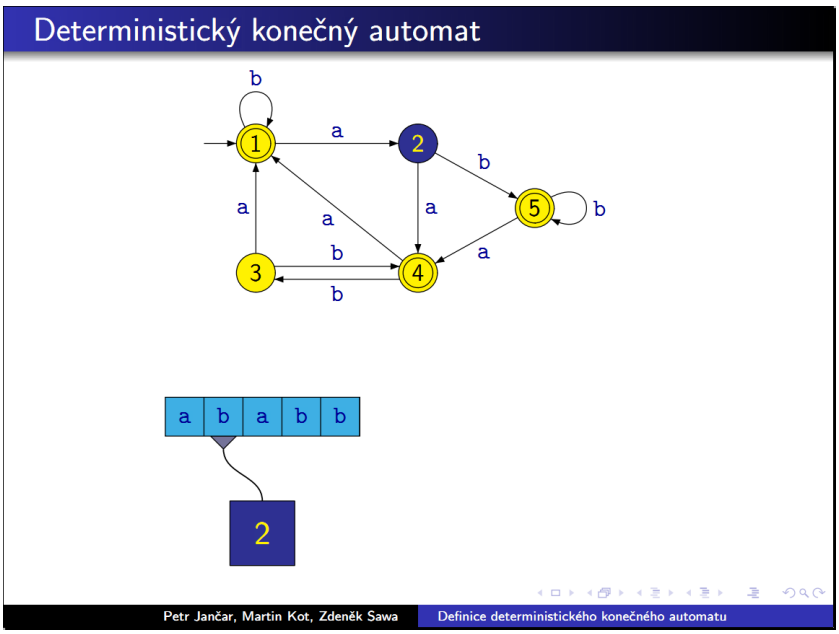


OBRÁZEK 14 – VÝSLEDNÉ SCHÉMA

3.2. A-DEFINICE_DFA.PDF

V prezentaci [10] je velmi přehledným způsobem definován a popsán deterministický konečný automat, jsou zde srozumitelně vysvětleny základní pojmy související s deterministickým konečným automatem a na třech příkladech je popsáno jeho fungování.

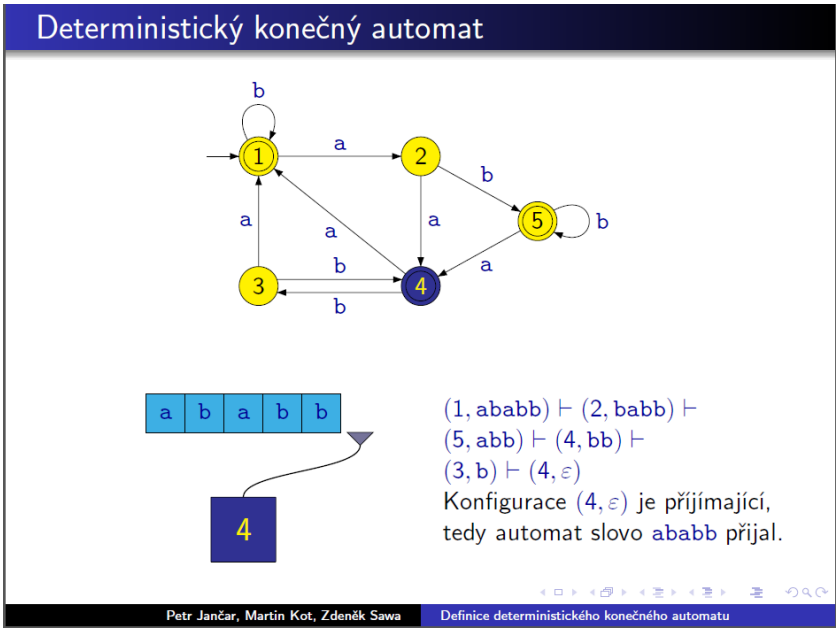
První „animace“ ukazuje, jak automat postupně načítá slovo. Čtecí hlava automatu se postupně přemísťuje zleva. Ve čtverci, který znázorňuje automat, se mění hodnota podle stavu, ve kterém se automat právě nachází a ve stavovém diagramu se postupně mění barva aktuálního stavu ze žluté na modrou.



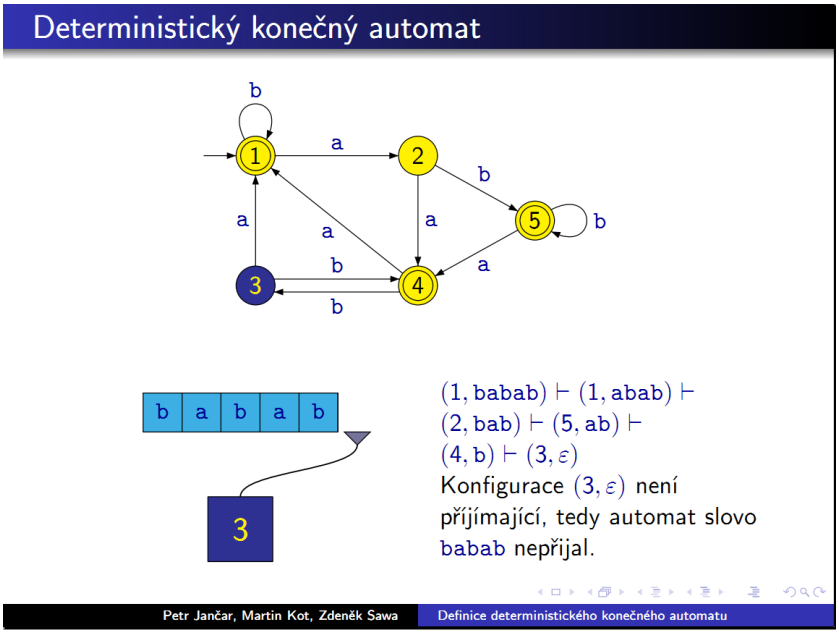
OBRÁZEK 15 – NAČÍTÁNÍ SLOVA AUTOMATEM

Dále je v prezentaci vysvětlen pojem konfigurace konečného automatu a následující dvě „animace“ ukazují rozdíl mezi slovem, které konečný deterministický automat přijímá (rozpoznává, akceptuje) a které ne.

Zápis *(1, ababb) ⊢ (2, babb)* znamená, že se ze stavu *1* po načtení znaku *a* dostaneme do stavu *2* atd.



OBRÁZEK 16 – PŘIJÍMACÍ KONFIGURACE



OBRÁZEK 17 – KONFIGURACE, KTERÁ NENÍ PŘIJÍMACÍ

4. WEBOVÉ ROZHRANÍ

Protože žádné z nalezených řešení nenabízí studentům možnost vygenerování schématu z popisu jejich KA, přináší jim tuto možnost dále popisovaná webová aplikace. Kromě generování stavových diagramů má uživatel možnost kontroly svých návrhů.

První kontrola při zadávání s pomocí dialogu probíhá po zadání počtu stavů a znaků ve vstupní abecedě. Druhá kontrola se aktivuje po zadání přechodové funkce do vygenerované tabulky. Podobná kontrola probíhá i při načítání dat ze souboru.

Po formálně správném zadání má uživatel možnost vyzkoušet si, jak jeho KA načítá zadávaná slova.

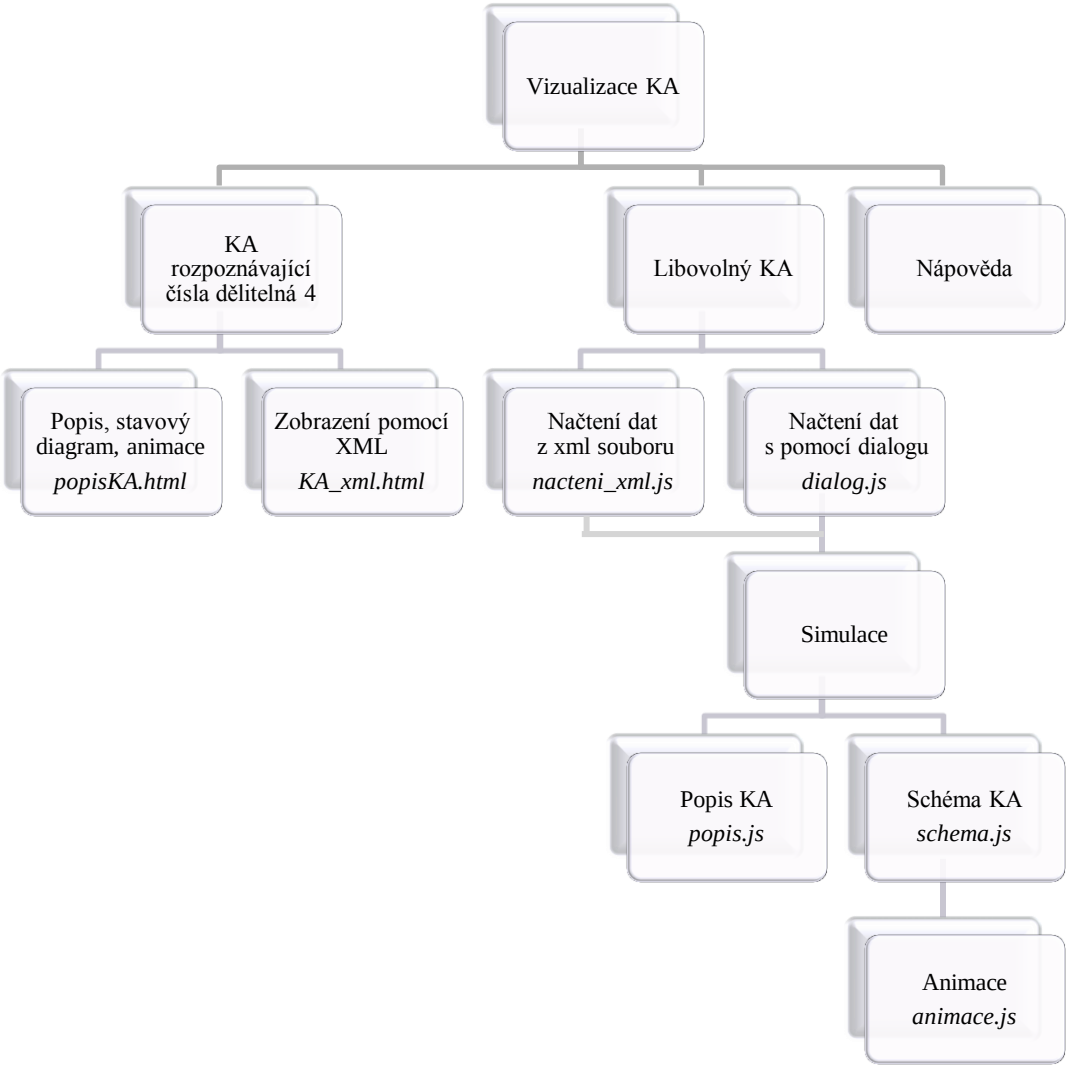
Jazykem pro implementaci je JavaScript, pro popis automatu v souboru byl vybrán jazyk XML, dialekt popsaný v kapitole 2.1. a stavové diagramy jsou popsány jazykem SVG.

Aplikace je funkční v prohlížeči Mozilla Firefox 4.0.

4.1. ÚVODNÍ STRÁNKA

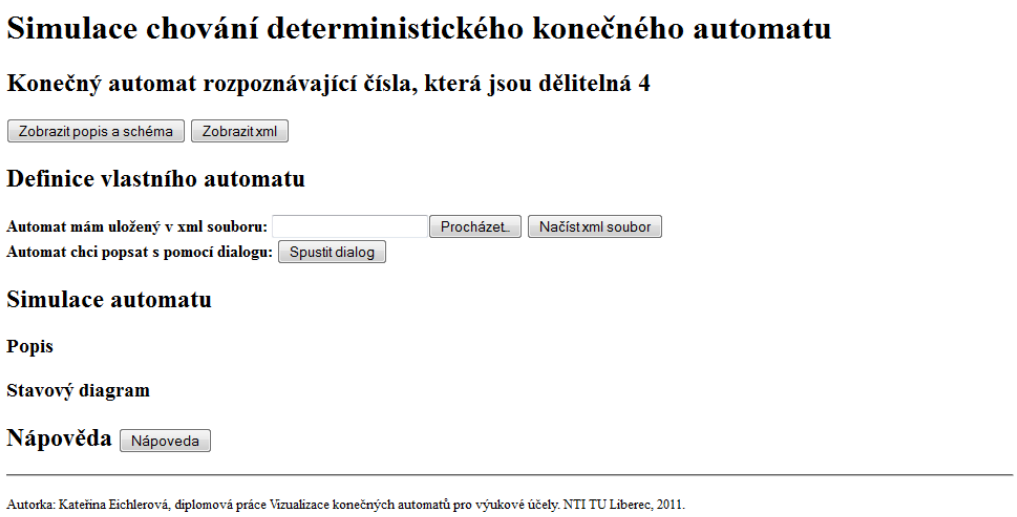
Úvodní stránka *index.html* je rozdělena na tři hlavní části. První oddíl je věnován zobrazení a popisu konečného deterministického automatu, který rozpoznává přirozená čísla dělitelná čtyřmi. Druhá partie umožňuje zadat uživateli parametry libovolného konečného automatu, nechat si jej zobrazit a případně vyzkoušet, zda jím definovaný konečný automat rozpozná zadané slovo. V případě nejasností je možno použít nápovědu ve třetí části stránky.

Schéma úvodní stránky *index.html*:



OBRÁZEK 18 – SCHÉMA ÚVODNÍ STRÁNKY

Zobrazení úvodní stránky v prohlížeči:



OBRÁZEK 19 – ÚVODNÍ STRÁNKA

4.2. KONEČNÝ DETERMINISTICKÝ AUTOMAT NAD ABECEDOU PŘIROZENÝCH ČÍSEL, KTERÝ ROZPOZNÁVÁ ČÍSLA DĚLITELNÁ ČTYŘMI

Zobrazit popis a schéma

Klepnutím na tlačítko *Zobrazit popis a schéma* se otevře nová stránka *popisKA.html*, která obsahuje popis automatu (rozbor automatu v kapitole 2.2) a jeho znázornění stavovým diagramem. Funkčnost tlačítka zajišťuje metoda *onclick="window.location.href=`popisKA.html`"*.

Stavový diagram je popsán jazykem *svg* přímo ve zdrojovém kódu stránky *popisKA.html*.

Nad stavovým diagramem je pole pro zadání vstupního slova a tlačítko *Animovat* pro spuštění animace přechodů mezi jednotlivými stavy. Animace je určena funkcemi *krok_del4()*, *onZnak1click()*, *onZnak2click()* a *onZnak3click()* ve scriptu stránky *popisKA.html*.

Po spuštění animace je zadané slovo načteno, jednotlivé znaky jsou porovnány se vstupní abecedou (tedy se znaky *L*, *D*, *S*). Pro symboly, které

jsou prvky vstupní abecedy, je elementu *svg* přidán potomek *text*, jehož *textContent* odpovídá danému znaku abecedy. Každému textovému elementu je přiřazena animovací funkce *onZnak1click()*, nebo *onZnak2click()*, nebo *onZnak3click()*.

Tato funkce vyhodnotí, který ze stavů je aktivní (má šedou výplň) a podle definované přechodové funkce najde nový stav. Původně aktivnímu stavu je změněna výplň z *lightgray* na *none* a nově aktivnímu stavu je změněna výplň z *none* na *lightgray*.

Na znaky můžeme klepnout i opakovaně. Po opakovaném klepnutí na určitý znak se znovu spustí animace, která je symbolu přidělena. Jestliže tedy zadáme do vstupního pole posloupnost znaků např.: *LSCXPSVDOKSL*, pod polem pro zadávání vstupního slova se zobrazí posloupnost: *LSSDSL* a postupným „klikáním“ můžeme simulovat přechody mezi jednotlivými stavy.

Pokud bychom do vstupního pole zadali a následně načetli pouze symboly *D, S, L*, mohli bychom načíst stejné slovo jako v předchozím případě – postupně bychom klepli na *L, S, S, D, S* a *L*.

Na stránce *popisKA.html* je tlačítko *Skrýt popis a schéma*, které nás opět pomocí metody *onclick=“window.location.href=`index.html`“* vrací na úvodní stránku.

Zobrazit xml

Otevře novou stránku, ve které je automat popsán jazykem XML. Stránka je také doplněna tlačítkem pro návrat na úvodní stránku

Stránka *popisKA.html*

Skrýt popis a schéma

Název KA:	**dělitelnost 4**
Množina stavů:	**dělí, sudá, lichá**
Vstupní abeceda:	**L** ={1, 3, 5, 7, 9} , **S** ={2, 6} , **D** ={0, 4, 8}
Počáteční stav:	**dělí**
Rozpoznávací stav:	**dělí**

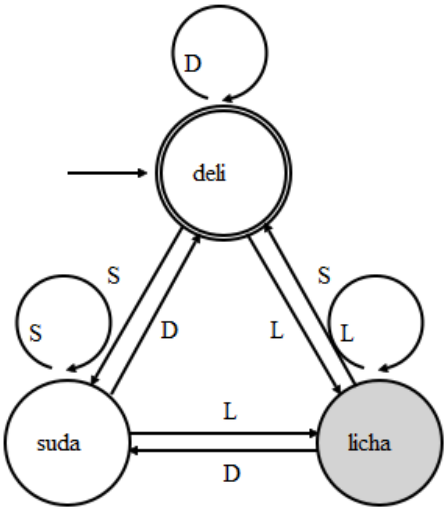
Přechodová funkce zadaná tabulkou:

	D	**S**	**L**
dělí	dělí	sudá	lichá
sudá	dělí	sudá	lichá
lichá	sudá	dělí	lichá

Stavový diagram:

Slovo pro automat: DDLSSLXCLS Animovat

D D L S S L L S



OBRÁZEK 20 – ZOBRAZENÍ POPISU A STAVOVÉHO DIAGRAMU

4.3. ZOBECNĚNÍ PRO LIBOVOLNÝ DETERMINISTICKÝ KONEČNÝ AUTOMAT

4.3.1. NAČTENÍ DAT Z XML SOUBORU

Uživatel určí adresu souboru s koncovkou *.xml*, ve kterém má uložen konečný automat (pokud možno správně) popsaný jazykem XML, dialektem podle návrhu autorky (viz kapitola 2.1).

Po stisknutí tlačítka *Načíst xml soubor* se spustí funkce *importXML(file)*, která je uložena v souboru *nacteni_xml.js*. Tato funkce se pokusí zadaný soubor otevřít. Pokud bude pokus úspěšný, bude spuštěna funkce *WriteXML()*, která načte data a vyhodnotí je. Při bezchybném zadání bude v části *Simulace automatu* zadaný automat popsán a bude zobrazen jeho stavový diagram.

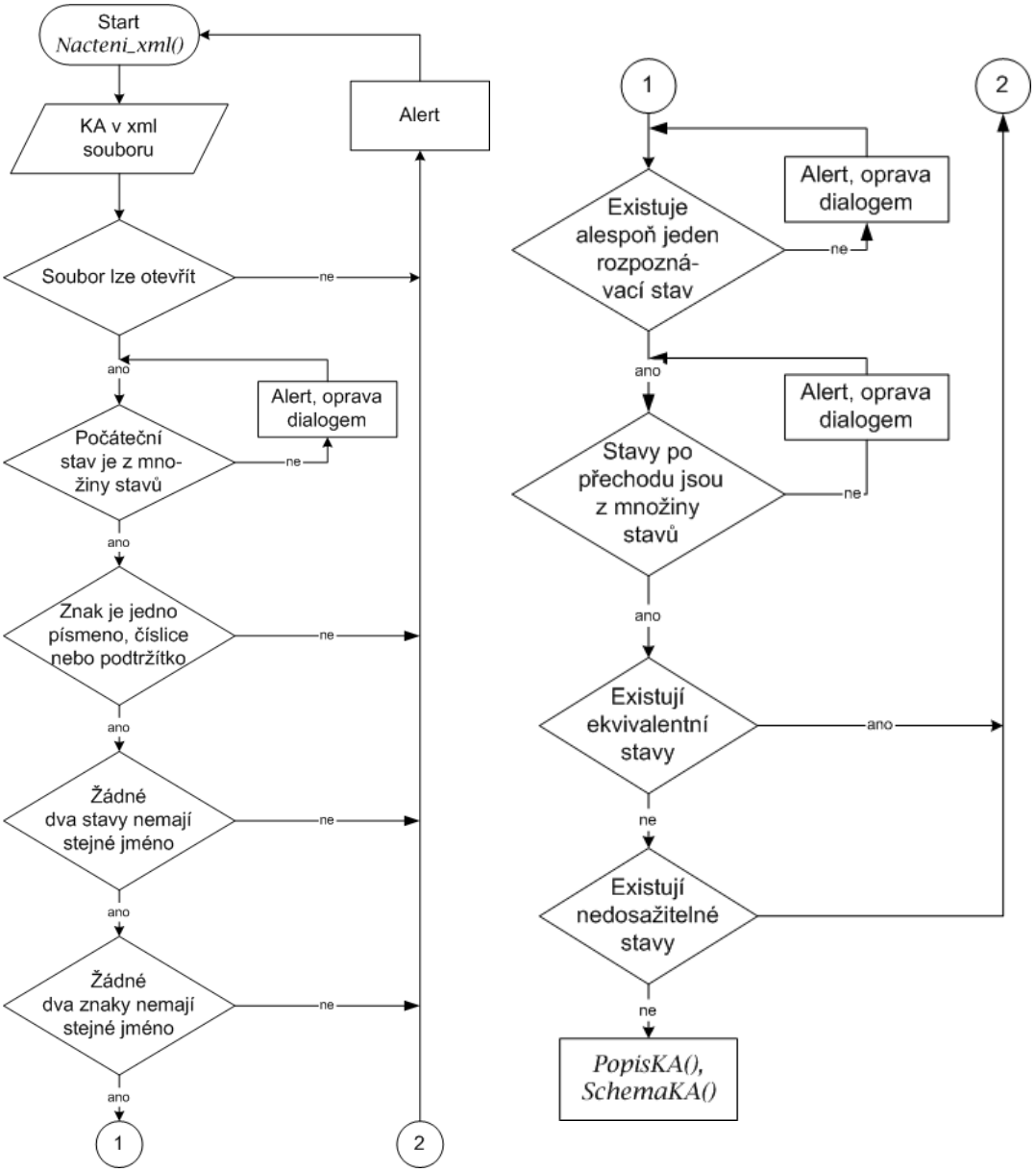
Funkce WriteXML() kontroluje, zda:

- je znak jeden prvek typu text – písmeno, číslice nebo podtržítko; porovnání s regulárním výrazem ^\w{1}$.
- se pro každý ze stavů neopakují názvy znaků
- se neopakují názvy stavů

Při výskytu alespoň jedné takové chyby je uživatel podrobně informován, kde se jaká chyba vyskytla. Každou z těchto chyb je potřeba opravit ve vstupním souboru a soubor znovu načíst.

- je zadán počáteční stav
- není po přechodu určen jiný stav, než je hodnota Atributu *Id* u jednotlivých stavů
- stavy nejsou ekvivalentní
- existuje alespoň jeden rozpoznávací stav

I na výskyt těchto chyb je uživatel upozorněn, ale může je opravit s pomocí dialogu. Oprava se ale do souboru nezapíše.



OBRÁZEK 21 – SCHÉMA NAČTENÍ XML

4.3.2. NAČTENÍ DAT S POMOCÍ DIALOGU

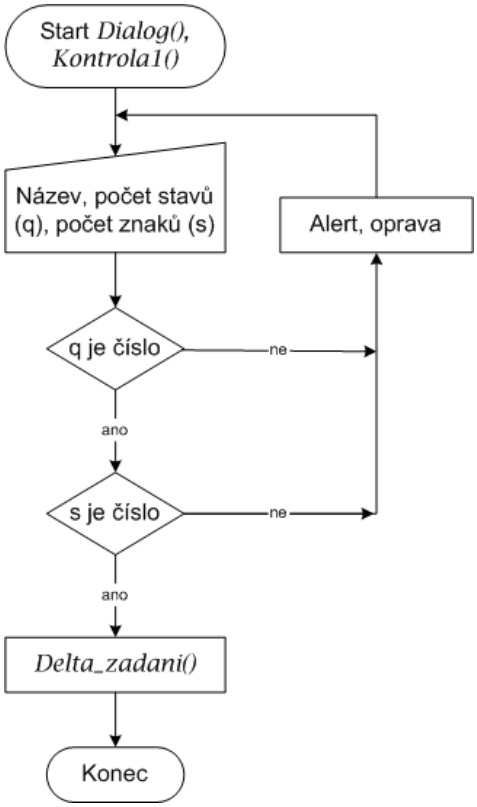
Pokud uživatel z nějakého důvodu nevyužije možnost načtení údajů o konečném deterministickém automatu z xml souboru, může k zadání využít dialog. Dialog je v rámci stránky rozdělen do dvou oddílů pomocí tagu <*div*>.

Po klepnutí na tlačítko *Spustit dialog* je aktivována funkce *dialog()*, která zviditelní oddíl s první tabulkou a druhý oddíl schová, pro případ, že by byl dialog spouštěn opakovaně. Funkce také ověří, zda již dříve nebyla tabulka vytvořena. Pokud již tabulka existuje, bude odstraněna a následně bude vygenerována tabulka nová, do které uživatel doplní název automatu, počet jeho stavů a počet znaků vstupní abecedy. Jednotlivé řádky tabulky jsou generovány jako potomci tabulky *dialog_tab1* a buňky jsou potomky řádků.

Po zadání a klepnutí na tlačítko *Načíst data* je kontrolováno, zda je počet stavů a počet znaků kladné celé číslo. Tato kontrola je prováděna porovnáním hodnoty v textovém poli s regulárním výrazem *[1-9][0-9]?*.

Tabulka *dialog_tab1*:

OBRÁZEK 22 – NAČTENÍ DAT S POMOCÍ DIALOGU



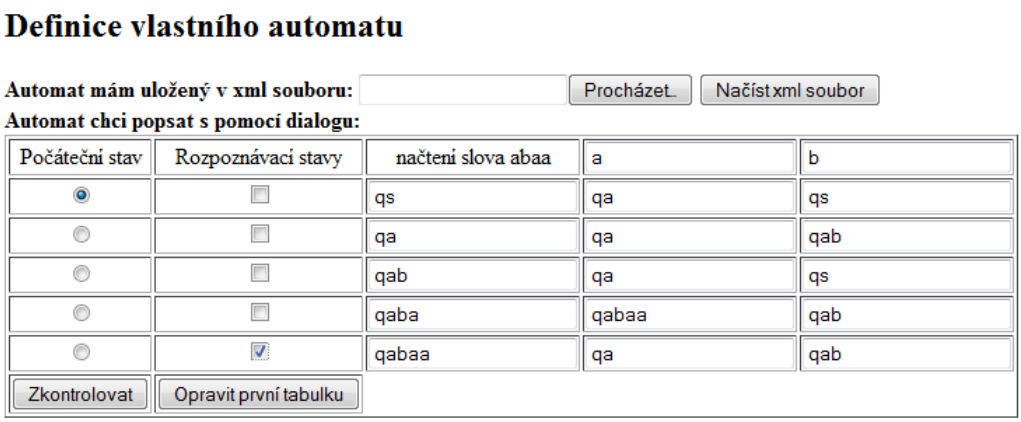
OBRÁZEK 23 – SCHÉMA NAČTENÍ DAT S POMOCÍ DIALOGU

Po správném zadání základních dat se spustí funkce *delta_zadani()*, která skryje první a zobrazí druhou tabulku. Funkce zkontroluje, zda již tabulka existuje – zda má tabulka *dialog_tab2* nějaké potomky. V kladném případě budou všichni odstraněni. Dále je generována tabulka přechodové funkce podle zadaného počtu stavů a počtu znaků – počet sloupců je o jednu větší než počet znaků a počet řádků je o jednu větší než počet stavů. Každý řádek je vytvořen jako potomek tabulky *dialog_tab2* a každá buňka jako potomek jednoho z řádků. V tabulce přechodové funkce uživatel označí stav počáteční a rozpoznávací stavy a vyplní názvy jednotlivých stavů, znaky vstupní abecedy a stavy po přechodu.

Pokud si uživatel uvědomí chybu v zadání počtu stavů nebo znaků, může se pomocí tlačítka *Opravit první tabulku* vrátit zpět a zadat jiné údaje. Toto tlačítko znovu volá funkci *dialog()*.

Po zadání údajů a klepnutí na tlačítko *Zkontrolovat* se spustí funkce *kontrola2()*.

Tabulka pro zadání přechodové funkce *dialog_tab2*:



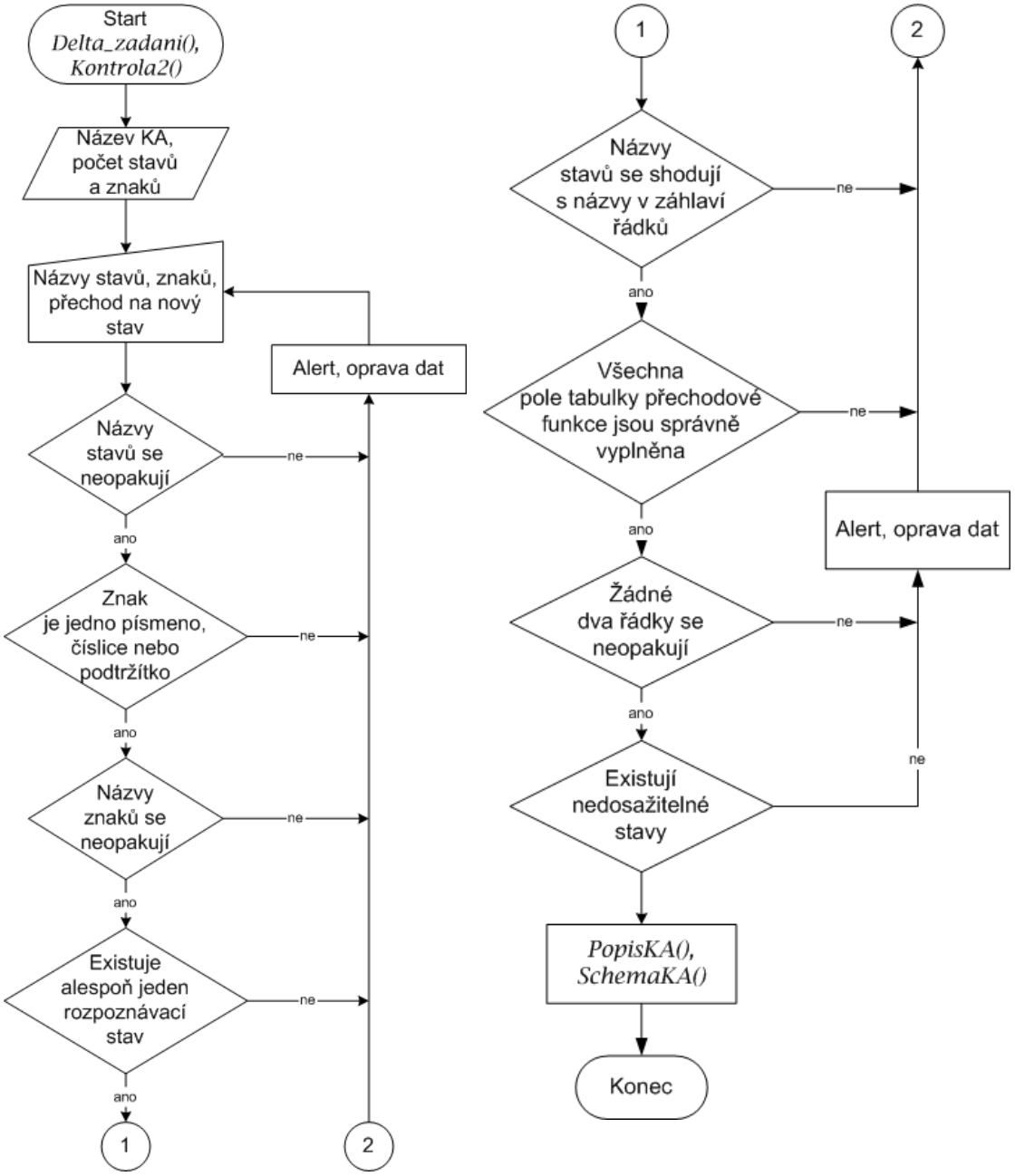
OBRÁZEK 24 – TABULKA PRO DEFINICI PŘECHODOVÉ FUNKCE

Funkce *kontrola2()* ověřuje, zda:

- se názvy stavů a znaků neopakují
- je znak právě jedna číslice, písmeno nebo podtržítko
- existuje alespoň jeden rozpoznávací stav
- jsou všechna pole tabulky správně vyplněna (nejsou prázdná, nebo neobsahují názvy stavů, které nejsou v záhlaví řádků)
- se žádné dva řádky neopakují
- tabulka neobsahuje žádný nedosažitelný stav

Při výskytu některé z těchto chyb je uživatel informován o tom, kde se jaká chyba vyskytla a je vyzván k opravě. Po novém zadání je provedena nová kontrola.

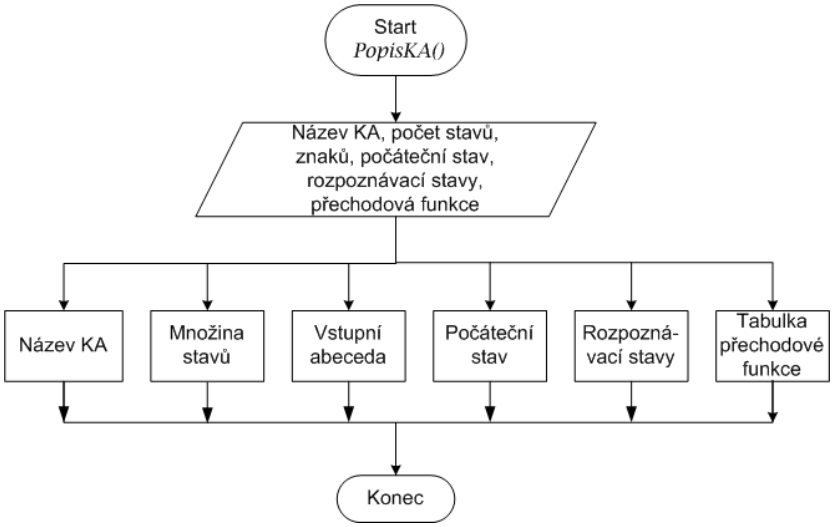
Kontrola zadání znaku se provádí porovnáním načteného slova s regulárním výrazem *^\w{1}$*, tedy zda je na začátku textový řetězec (písmeno, číslice nebo podtržítko) o délce právě jeden znak.



OBRÁZEK 25 – SCHÉMA DRUHÉ KONTROLY

4.3.3. POPIS KA

Po načtení dat od uživatele se v další sekci stránky objeví přehledný výpis názvu KA, množiny stavů, vstupní abecedy, počátečního stavu a rozpoznávacích stavů. Přechodová funkce je popsána tabulkou.



OBRÁZEK 26 – SCHÉMA POPISU KA

Simulace automatu

Popis

Název KA: čísla dělitelná 4

Množina stavů: sudá lichá dělí

Vstupní abeceda: S L D

Počáteční stav: dělí

Rozpoznávací stavy: dělí

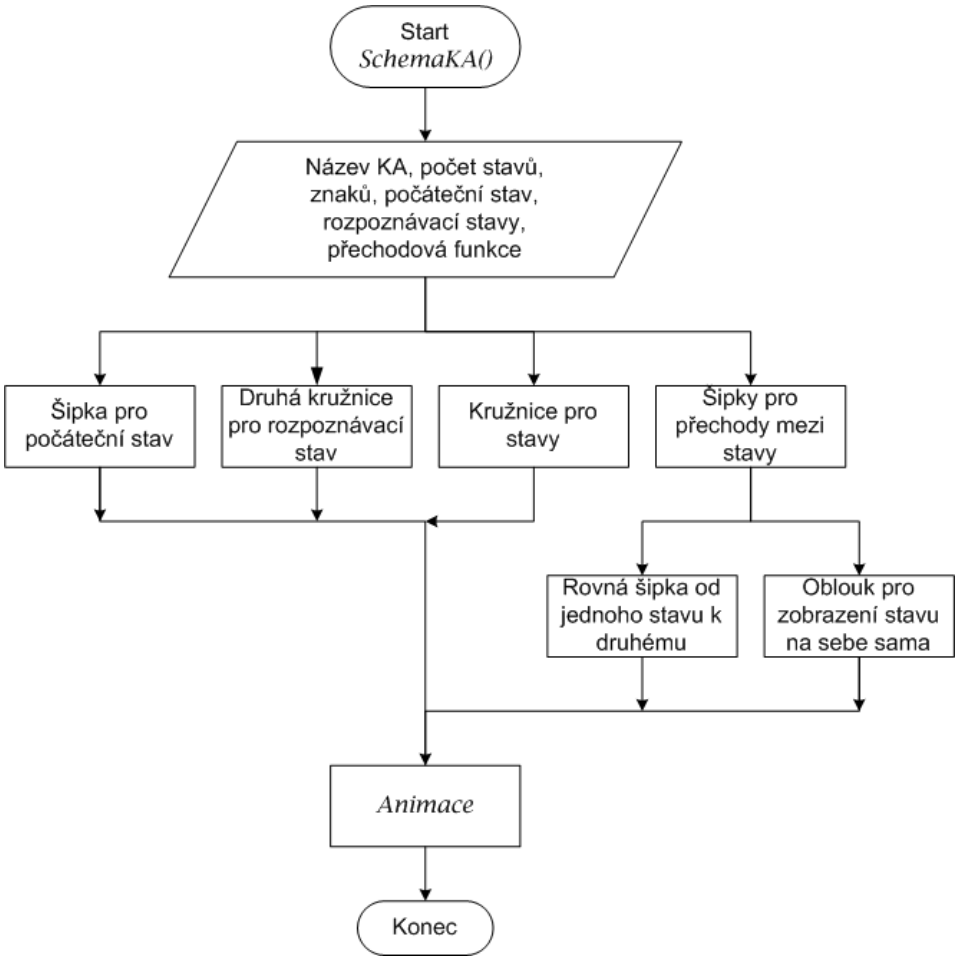
Přechodová funkce zadaná tabulkou:

	S	L	D
sudá	sudá	lichá	dělí
lichá	dělí	lichá	sudá
dělí	sudá	lichá	dělí

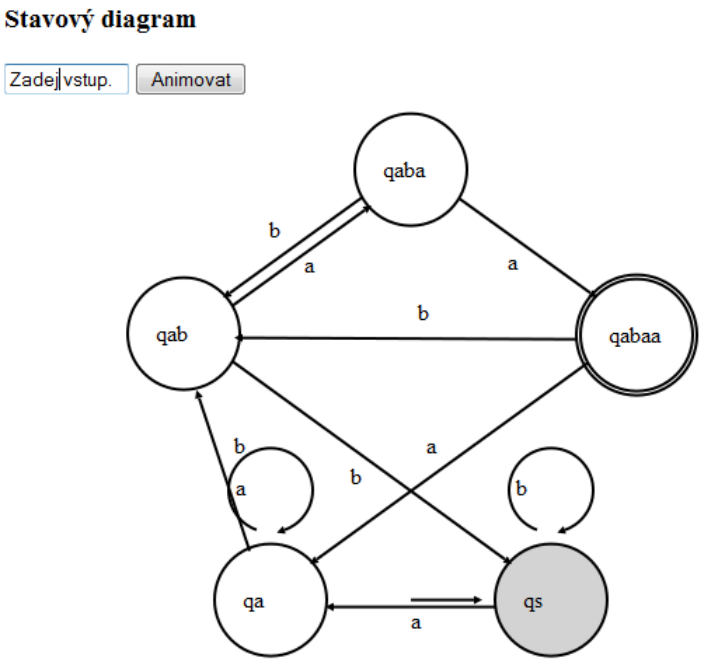
OBRÁZEK 27 – POPIS KA

4.3.4. SCHÉMA KA

V této části stránky se po načtení dat zobrazí stavový diagram zadaného konečného automatu. Stavy jsou znázorněny kruhy, počáteční stav má světle šedou výplň a směřuje k němu krátká šipka, rozpoznávací stavy jsou označeny dvojitou kružnicí. Přechody zobrazují šipky od původního stavu k novému, v případě, že se stav zobrazí sám na sebe, je přechod znázorněn obloukem se šipkou.



OBRÁZEK 28 – SCHÉMA PRO ZOBRAZENÍ STAVOVÉHO DIAGRAMU



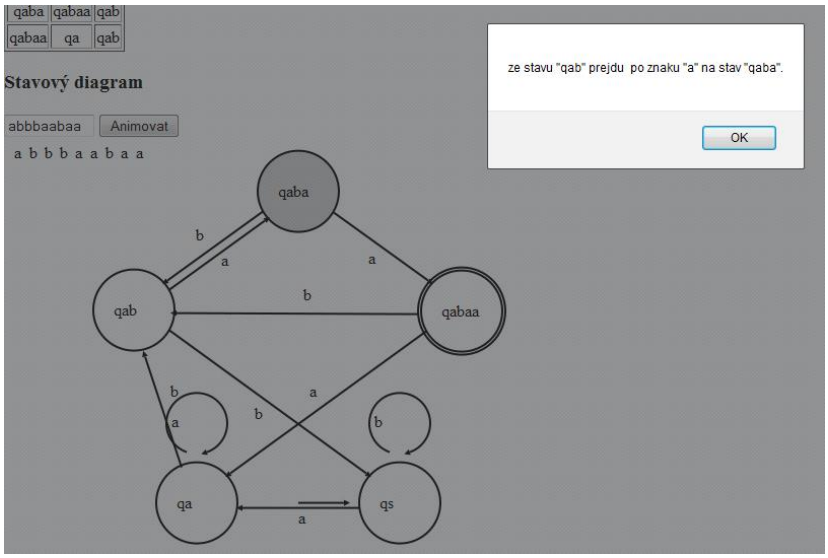
OBRÁZEK 29 – STAVOVÝ DIAGRAM; KA ROZPOZNÁVAJÍCÍ SLOVA, KTERÁ KONČÍ NA *abaa*

4.3.5. ANIMACE

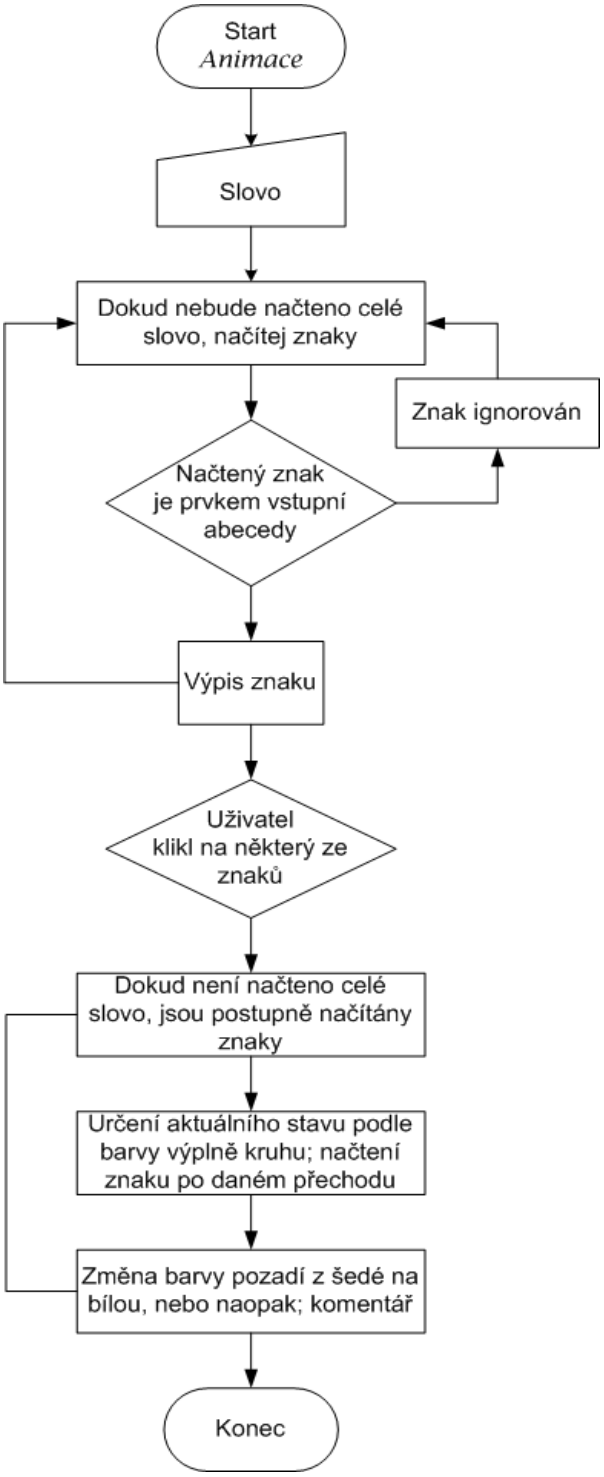
Po zadání slova do vstupního pole a stisknutí tlačítka *Animovat* je aktivována funkce *krok()*, která znak po znaku načítá uživatelem zadané vstupní slovo. Vedle stavového diagramu z něj zobrazí pouze ty znaky, které jsou prvky vstupní abecedy. Znaky jsou, stejně jako při animaci automatu rozpoznávajícího čísla dělitelná čtyřmi, zobrazeny jako potomci elementu *svg*, který popisuje stavový diagram. Každý textový potomek má nastavenu vlastnost *textContent* na hodnotu daného znaku a je mu přiřazena funkce *onZnakClick_krok*, která zajišťuje změnu barvy aktivního stavu a zobrazení komentáře. Obdobně funguje i fukce *krok_xml()*, která je spuštěna při animaci v případě, že údaje o DKA byly načteny ze souboru.

Po načtení znaku ze vstupní abecedy, který je zobrazen vedle diagramu dojde k jeho porovnání se vstupní abecedou a určení, o který ze znaků se jedná. Dále skript určí, který ze stavů je aktivní (jeho výplň je šedá) a vyhledá nový stav, podle zadané přechodové funkce. Původně aktivnímu stavu je odebrána šedá výplň, která je přidělena nově aktivnímu stavu. Cyklus se opakuje, dokud není načteno celé slovo.

Pro spuštění animace stačí klepnout na jakýkoliv zobrazený znak.



OBRÁZEK 30 – KOMENTOVANÁ ANIMACE PŘECHODŮ MEZI STAVY



OBRÁZEK 31 – SCHÉMA ANIMACE

5. ZÁVĚR

Cílem diplomové práce bylo vytvořit webovou aplikaci, kterou bude možno použít při studiu konečných automatů. Věřím, že se mi podařilo zadání naplnit. Výsledkem práce je aplikace, která umožňuje online zadat libovolný konečný automat a provést simulaci jeho chování. Automat je možné zadávat i s pomocí jeho definice v XML.

SEZNAM PRAMENŮ

[1] AUBURN, RJ, et al. *W3C* [online]. 2005 [cit. 2011-07-02]. State Chart XML (SCXML): State Machine Notation for Control Abstraction 1.0. Dostupné z WWW: <http://www.w3.org/TR/2005/WD-scxml-20050705/>

[2] BARNETT, Jim , et al. ***W3C*** [online]. 2010 [cit. 2011-02-07]. State Chart XML (SCXML): State Machine Notation for Control Abstraction. Dostupné z WWW: <http://www.w3.org/TR/2010/WD-scxml-20101216/>

[3] BÜRGER, Fabian; WINNEKENS, Kai. *Arcor* [online]. 2009 [cit. 2011-02-07]. OOP in Java Project Work - A simulator for finite state automatons. Dostupné z WWW: <http://home.arcor.de/kai.w1986/dfasimulator/>

[4] CLARK, James; DEROSE, Steve. *W3C* [online]. 1999 [cit. 2011-02-07]. XML Path Language (XPath). Dostupné z WWW: <http://www.w3.org/TR/1999/REC-xpath-19991116/>

[5] EISENBERG, David John . *SVG Esentials*. Liverpool : O'Reilly Media, 2002. 368 s. ISBN 0-596-00223-8

[6] GRIMMICH, Šimon . ***Tvorba webu*** [online]. 2008 [cit. 2011-03-29]. JavaScript: Události. Dostupné z WWW: <http://www.tvorba-webu.cz/javascript/udalosti.php>

[7] GRIMMICH, Šimon . ***Tvorba webu*** [online]. 2008 [cit. 2011-03-29]. JavaScript: String. Dostupné z WWW: http://www.tvorba-webu.cz/javascript/string.php

[8] GRIMMICH, Šimon . ***Tvorba webu*** [online]. 2008 [cit. 2011-03-29]. JavaScript: Regulární výrazy. Dostupné z WWW: <http://www.tvorba-webu.cz/javascript/regularni_vyrazy.php>

[9] HEROUT, Pavel. ***XSLT 2.0 a SVG prakticky***. České Budějovice : KOPP, 2010. 296 s. ISBN 978-80-7232-406-4

[10] JANČAR, Petr; KOT, Martin; SAWA, Zdeněk. Definice deterministického konečného automatu. In ***Deterministický konečný automat*** [online]. Ostrava : Vysoká škola báňská - Technická univerzita Ostrava, 2008 [cit. 2011-02-27]. Dostupné z WWW: <http://www.cs.vsb.cz/kot/anim/a-definice_dfa.pdf>

[11] JANOVSKÝ, Dušan. *Jak psát web* [online]. 2005 [cit. 2011-02-22]. Dostupné z WWW: <http://www.jakpsatweb.cz/>. ISSN 1801-0458

[12] KOSEK, Jiří. *Xml pro každého* [online]. 1. vydání. Praha : Grada Publishing, 2000 [cit. 2011-02-09]. Dostupné z WWW: <http://www.kosek.cz/xml/xmlprokazdeho.pdf>. ISBN 80-7169-860-1.

[13] KOSEK, Jiří. XML schémata. In *XML schémata* [online]. [s. l.] : [s. n.], 18. 8. 2005 [cit. 2011-02-04]. Dostupné z WWW: <http://www.kosek.cz/xml/schema/xmlschema.pdf>

[14] KOSEK, Jiří. *XSLT v příkladech* [online]. 11. 9. 2003 [cit. 2011-02-18]. Dostupné z WWW: <http://www.kosek.cz/xml/xslt/index.html>KOSEK, Jiří

[15] NIC, Miloslav. *Zvon.org* [online]. 2000 [cit. 2011-02-15]. XPath 1.0 Tutorial @ZVON.org. Dostupné z WWW: <http://zvon.org/comp/r/tut-XPath_1.html>

[16] The Vaucanson Group. ***Finite State Machine XML (FSMXML) Specifications*** [online]. [s.l.] : [s.n.], září 2008 [cit. 2011-02-08]. Dostupné z WWW: <http://www.lrde.epita.fr/dload/vaucanson/techrep/fsmxml-0.5.spec.pdf>

[17] TIDWELL, Doug. *IBM* [online]. leden 2000, březen 2001 [cit. 2011-02-13]. DeveloperWorks : XML : Transforming XML into SVG. Dostupné z WWW: <http://www.ling.helsinki.fi/kit/2006k/clt 232/tutorials/XMLtoSVG.html>

[18] TIŠNOVSKÝ, Pavel. ***Root*** [online]. 2008 [cit. 2011-01-03]. Dostupné z WWW: <http://www.root.cz/serialy/graficke-formaty/>

[19] ***W3schools*** [online]. 2010 [cit. 2011-03-29]. HTML DOM Form Object. Dostupné z WWW: <http://w3s.ir/jsref/dom_obj_form.asp.htm>

[20] ***W3schools*** [online]. 2010 [cit. 2011-03-29]. HTML DOM Text Object. Dostupné z WWW: <http://w3s.ir/jsref/dom_obj_text.asp.htm>

[21] ZONER software, a.s. ***Interval.cz*** [online]. 2005 [cit. 2011-03-29]. Dostupné z WWW: <http://interval.cz/programovani/javascript-ajax/>

[22] ZONER software, a.s. ***Interval.cz*** [online]. 2006 [cit. 2011-01-3]. Dostupné z WWW: <http://interval.cz/webdesign/webova-grafika/>

6. PŘÍLOHA – ZDROJOVÉ KÓDY APLIKACE

6.1. INDEX.HTML

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Deterministický konečný automat</title>
  </head>
  <body>
     <h1>Simulace chování deterministického konečného
        automatu</h1>
     <h2>Konečný automat rozpoznávající čísla, která jsou
        dělitelná 4</h2>
     <input type="button" id="zs" value="Zobrazit popis a
        schéma" onclick="window.location.href='popisKA.html'"/>
     <input type="button" id="s_xml" value="Zobrazit xml"
        onclick="window.location.href='KA_xml.html'"/>

     <h2>Definice vlastního automatu</h2>
     <b>Automat mám uložený v xml souboru:</b>
     <input id="soubor" type="file"/>  <input type="button"
        value="Načíst xml soubor"
        onclick="importXML(document.getElementById('soubor').va
        lue)"/><br/>
     <b>Automat chci popsat s pomocí dialogu:</b>
     <input type="button" id="sd" value="Spustit dialog"
        onclick="dialog()"/>
    <div id="z_udaje" hidden='true'>
      <table id="dialog_tab1" border="1"></table>
    </div>
    <div id="vyplneni">
      <table id="dialog_tab2" border="1"></table>
    </div>

    <h2>Simulace automatu</h2>
    <h3>Popis</h3>
    <div id="popis"></div>
    <div id="popis_tab" hidden="true">
      <table id="tabKA" border="1"></table>
    </div>

    <h3>Stavový diagram</h3>
    <div id="KA_animace" hidden='true'>
      <input type="text" id="vstup" size="10" value="Zadej
        vstup."/>
      <input type="button" value="Animovat" onclick="krok()"/>
    </div>
    <div id="schema"></div>
```

```
    <h2>Nápověda
    <input type="button" id="napoveda" value="Nápověda"
        onclick="napoveda()"> </h2>

    <hr/>
    <p><small>Autorka: Kateřina Eichlerová, diplomová práce
        Vizualizace
        konečných automatů pro výukové účely. NTI TU Liberec,
        2011.</small></p>

    <script src="delitelnost4.js"></script>
    <script src="dialog.js"></script>
    <script src="nacteni_xml.js"></script>
    <script src="popisKA.js"></script>
    <script src="schemaKA.js"></script>
    <script src="animace.js"></script>

  </body>
</html>
```

6.2. KONEČNÝ AUTOMAT ROZPOZNÁVAJÍCÍ ČÍSLA DĚLITELNÁ ČTYŘMI

6.2.1. POPISKA.HTML

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Popis KA</title>
  </head>
  <body>
    <input type="button" id="s_s" value="Skrýt popis a schéma"
       onclick="window.location.href='index_1.html'"/>
    <p>
      <table id="del4_popis" border="1">
        <tr>
          <td>Název KA: </td>
          <td><b>dělitelnost 4</b></td>
        </tr>
        <tr>
          <td>Množina stavů: </td>
          <td><b>dělí, sudá, lichá</b></td>
        </tr>
        <tr>
          <td>Vstupní abeceda: </td>
          <td><b>L</b> <small> ={1, 3, 5, 7, 9} </small>,<b> S
       </b><small> ={2, 6} </small>, <b>D</b>
       <small> ={0, 4, 8} </small></td>
        </tr>
```

```
    <tr>
      <td>Počáteční stav: </td>
      <td><b>dělí</b></td>
    </tr>
    <tr>
      <td>Rozpoznávací stav: </td>
      <td><b>dělí</b></td>
    </tr>
  </table>
</p>
<p><b>Přechodová funkce zadaná tabulkou:</b></p>
<table id="del4_delta" border="1">
  <tr>
   <td></td>
    <td><b>D</b></td>
    <td><b>S</b></td>
    <td><b>L</b></td>
  </tr>
  <tr>
    <td><b>dělí</b></td>
    <td>dělí</td>
    <td>sudá</td>
    <td>lichá</td>
  </tr>
  <tr>
    <td><b>sudá</b></td>
    <td>dělí</td>
    <td>sudá</td>
    <td>lichá</td>
  </tr>
  <tr>
    <td><b>lichá</b></td>
    <td>sudá</td>
    <td>dělí</td>
    <td>lichá</td>
  </tr>
</table>
<p><b>Stavový diagram:</b></p>
<p>Slovo pro automat: <input type="text" id="del4_vstup"
   size="10" value="Zadej vstup."/>
<input type="button" value="Animovat"
   onclick="krok_del4()"/></p>

<svg xmlns="http://www.w3.org/2000/svg" id="svg"
   width="800px" height="360px">
  <circle id="stav1" fill="none" stroke="black" stroke-
   width="2" cx="600" cy="310" r="40">
    <animate id="blik1" attributeName="fill"
   from="lightgray" to="none" begin="indefinite"
   dur="300ms" fill="remove" />
  </circle>
  <text style="font-size:20" x="580" y="315">licha</text>
  <circle id="stav2" fill="none" stroke="black" stroke-
   width="2" cx="400" cy="310" r="40">
```

```
  <animate id="blik2" attributeName="fill"
 from="lightgray" to="none" begin="indefinite"
 dur="300ms" fill="remove" />
</circle>
<text style="font-size:20" x="380" y="315">suda</text>
<circle fill="none" stroke="black" stroke-width="2"
 cx="500" cy="137" r="43">
  <animate id="rsk23" attributeName="stroke" from="black"
 to="white" begin="indefinite" dur="300ms" fill="remove"
 />
</circle>
<line stroke="black" stroke-width="2" x1="400" y1="137"
 x2="445" y2="137" marker-end="url(#sipka)" />
<circle id="stav3" fill="none" stroke="black" stroke-
 width="2" cx="500" cy="137" r="40">
  <animate id="rsk13" attributeName="stroke" from="black"
 to="white" begin="indefinite" dur="300ms" fill="remove"
 />
  <animate id="blik3" attributeName="fill"
 from="lightgray" to="none" begin="indefinite"
 dur="300ms" fill="remove" />
</circle>
<text style="font-size:20" x="480" y="142">deli</text>
<path fill="none" stroke="black" stroke-width="2" d="M
 590 262 a 30 30 0 1 1 15 0" marker-end="url(#sipka)" />
<text x="575" y="245" style="font-size:20 font-
 family:Serif">L</text>
<line id="stav13" stroke="black" stroke-width="2"
 x1="585" y1="274" x2="529" y2="174" marker-
 end="url(#sipka)" />
<text x="560" y="208" style="font-size:20 font-
 family:Serif">S</text>
<line id="stav12" stroke="black" stroke-width="2"
 x1="560" y1="315" x2="445" y2="315" marker-
 end="url(#sipka)" />
<text x="500" y="335" style="font-size:20 font-
 family:Serif">D</text>
<line id="stav21" stroke="black" stroke-width="2"
 x1="440" y1="304" x2="555" y2="304" marker-
 end="url(#sipka)" />
<text x="500" y="295" style="font-size:20 font-
 family:Serif">L</text>
<path fill="none" stroke="black" stroke-width="2" d="M
 390 262 a 30 30 0 1 1 15 0" marker-end="url(#sipka)" />
<text x="375" y="245" style="font-size:20 font-
 family:Serif">S</text>
<line id="stav23" stroke="black" stroke-width="2"
 x1="428" y1="279" x2="482" y2="181" marker-
 end="url(#sipka)" />
<text x="460" y="243" style="font-size:20 font-
 family:Serif">D</text>
<line id="stav31" stroke="black" stroke-width="2"
 x1="515" y1="176" x2="572" y2="274" marker-
 end="url(#sipka)" />
```

```
<text x="530" y="243" style="font-size:20 font-
 family:Serif">L</text>
<line id="stav32" stroke="black" stroke-width="2"
 x1="474" y1="171" x2="418" y2="268" marker-
 end="url(#sipka)" />
<text x="425" y="208" style="font-size:20 font-
 family:Serif">S</text>
<path fill="none" stroke="black" stroke-width="2" d="M
 490 89 a 30 30 0 1 1 15 0" marker-end="url(#sipka)" />
<text x="475" y="72" style="font-size:20 font-
 family:Serif">D</text>
<defs>
  <marker id="sipka" viewBox="0 0 10 10" refX="0"
 refY="5" stroke="black" fill="black"
 markerUnits="strokeWidth" markerWidth="4"
 markerHeight="3" orient="auto">
    <path d="M 0 0 L 10 5 L 0 10 z" />
  </marker>
</defs>
<script type="text/ecmascript"> <![CDATA[
  function krok_del4()
  {
    alert("Zadané slovo se zobrazí vedle stavového
 diagramu.\nPo klepnutí na vybraný znak přejde KA do
 nového stavu.\nZnaky, které nejsou prvky vstupní abecedy
 budou ignorovány.");

    var znaky, i, zn;
    var slovo = document.getElementById('del4_vstup');
    znaky = slovo.value;

    zn = new Array();
    var x = 10;
    for (i=0; i<znaky.length; i++)
    {
      zn[i] = znaky.charAt(i);
      var svg = document.getElementById('svg');
      var text =
 document.createElementNS('http://www.w3.org/2000/svg',
 'text');

      if (zn[i] == "L")
      {
        text.setAttribute('onclick', 'onZnak1Click()');
        text.textContent=zn[i];
        x += 15;
        text.setAttribute('x', x);
        text.setAttribute('y', '20');
      }
      else if (zn[i] == "S")
      {
        text.setAttribute('onclick', 'onZnak2Click()');
        text.textContent=zn[i];
        x += 15;
```

```
      text.setAttribute('x', x);
      text.setAttribute('y', '20');
    }
    else if (zn[i] == "D")
    {
      text.setAttribute('onclick', 'onZnak3Click()');
      text.textContent=zn[i];
      x += 15;
      text.setAttribute('x', x);
      text.setAttribute('y', '20');
    }
    svg.appendChild(text);
  }
}

var q0 = document.getElementById("stav3");
q0.setAttribute("fill", "lightgray");

function onZnak1Click(event)
{
  var q1=document.getElementById("stav1");
  var q2=document.getElementById("stav2");
  var q3=document.getElementById("stav3");
  var anim1=document.getElementById("blik1");
  if (q1.getAttribute("fill") == ("lightgray"))
    anim1.beginElement();
  if (q2.getAttribute("fill") == ("lightgray"))
  {
    q2.setAttribute("fill", "none");
    q1.setAttribute("fill", "lightgray");
  }
  else
  {
    if (q3.getAttribute("fill") == ("lightgray"))
    {
      q3.setAttribute("fill", "none");
      q1.setAttribute("fill", "lightgray");
    }
  }
}

function onZnak2Click(event)
{
  var q1=document.getElementById("stav1");
  var q2=document.getElementById("stav2");
  var q3=document.getElementById("stav3");
  var anim2=document.getElementById("blik2");
  if (q2.getAttribute("fill") == ("lightgray"))
    anim2.beginElement();
  if (q1.getAttribute("fill") == ("lightgray"))
  {
    q1.setAttribute("fill", "none");
    q3.setAttribute("fill", "lightgray");
  }
```

```
          else
          {
            if (q3.getAttribute("fill") == ("lightgray"))
            {
              q3.setAttribute("fill", "none");
              q2.setAttribute("fill", "lightgray");
            }
          }
          if (q3.getAttribute("fill") == ("lightgray"))
          {
            var konec13=document.getElementById("rsk13");
            var konec23=document.getElementById("rsk23");
            konec13.beginElement();
            konec23.beginElement();
          }
        }

        function onZnak3Click(event)
        {
        var q1=document.getElementById("stav1");
        var q2=document.getElementById("stav2");
        var q3=document.getElementById("stav3");
        var anim3=document.getElementById("blik3");
        if (q3.getAttribute("fill") == ("lightgray"))
          anim3.beginElement();
        if (q1.getAttribute("fill") == ("lightgray"))
        {
          q1.setAttribute("fill", "none");
          q2.setAttribute("fill", "lightgray");
        }
        else
        {
          if (q2.getAttribute("fill") == ("lightgray"))
          {
            q2.setAttribute("fill", "none");
            q3.setAttribute("fill", "lightgray");
          }
        }
        if (q3.getAttribute("fill") == ("lightgray"))
        {
          var konec13=document.getElementById("rsk13");
          var konec23=document.getElementById("rsk23");
          konec13.beginElement();
          konec23.beginElement();
        }
      }
    ]]>
    </script></svg>
    <script src="animace.js"></script>
  </body>
</html>
```

6.2.2. KA_XML.HTML

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Popis KA v xml</title>
  </head>
  <body>
    <div id="del4_xml">
    <input type="button" id="s_xml" value="Skrýt xml"
       onclick="window.location.href='index_1.html'"/>
    <p>&lt;FA name="delitelnost_4" startstate="deli"&gt;<br/>
        &lt;state id="deli" accepting="true"&gt;<br/>
          &lt;transition symbol="L" target="licha"/&gt;<br/>
          &lt;transition symbol="S" target="suda"/&gt;<br/>
          &lt;transition symbol="D" target="deli"/&gt;<br/>
        &lt;/state&gt;<br/>
        &lt;state id="licha" accepting="false"&gt;<br/>
          &lt;transition symbol="L" target="licha"/&gt;<br/>
          &lt;transition symbol="S" target="deli"/&gt;<br/>
          &lt;transition symbol="D" target="suda"/&gt;<br/>
        &lt;/state&gt;<br/>
        &lt;state id="suda" accepting="false"&gt;<br/>
          &lt;transition symbol="L" target="licha"/&gt;<br/>
          &lt;transition symbol="S" target="suda"/&gt;<br/>
          &lt;transition symbol="D" target="deli"/&gt;<br/>
        &lt;/state&gt;<br/>
      &lt;/FA&gt;<br/> </p>
    </div>
  </body>
</html>
```

6.3. LIBOVOLNÝ KONEČNÝ AUTOMAT

6.3.1. NACTENI_XML.JS

```
var xmlDoc;

function importXML(file)
{
  typeof document.implementation != 'undefined';
  typeof document.implementation.createDocument != 'undefined';

  xmlDoc = document.implementation.createDocument('', '', null)
  xmlDoc.onload=WriteXML;
  xmlDoc.load(file);

  WriteXML();
}
```

```
function WriteXML()
{
  var chyba = 0;
// Zadani dat a kontrola delta tabulky.
  var FA = xmlDoc.getElementsByTagName('FA')[0];
  var state = FA.getElementsByTagName('state');
  var q=state.length+1;
  var trans = state[0].getElementsByTagName('transition');
  var s=trans.length+1;
  var nazev = FA.getAttribute("name");
  var q0 = FA.getAttribute("startstate");

// Vyplneni tabulky prechodove funkce delta a oznaceni
      rozpoznavacich stavu.
  var delta = new Array();
  var konec = new Array();
  var znak = new Array();
  var stav = new Array();
//znak je jedno pismeno nebo cislice
  var symbol = /^(\w){1}$/;

  for (i=0; i<q; i++)
  {
    delta[i] = new Array();
    konec[i] = new Array();
    for (j=0; j<s; j++)
    {
      if(i==0 && j!=0)
      {
        znak[j] = trans[j-1].getAttribute("symbol");
// znak je jedno písmeno, podtržítko nebo číslice
        if (!symbol.test(znak[j]))
        {
          alert("Znak"+i+" \""+znak[j]+"\" není jedno písmeno
      nebo jedna číslice!");
          chyba++;
        }
      }
      else if (i!=0 && j==0)
      {
        stav[i] = state[i-1].getAttribute("id");

// Kontrola2: jmena stavu v zahlavi tabulky se neopakuji.
        if (i>1)
        {
          for (k=i-1; k>0; k--)
          {
            if (stav[i] == stav[k])
            {
              alert("Stavy \"" +i+ "\" a \"" +k+ "\" maji
      stejny nazev!\n\""+stav[i]+"\"");
              chyba++;
            }
          }
```

```
        }
        konec[i] = state[i-1].getAttribute("accepting");
      }
      else if (i!=0 && j!=0 &&
      FA.getElementsByTagName('state')[i-1].getElementsByTag
      Name('transition')[j-1].getAttribute('target') != "null")
      {
        delta[i][j] = FA.getElementsByTagName('state')[i-1].get
      ElementsByTagName('transition')[j-1].getAttribute
      ('target');
// Kontrola2a: znaky se pro daný stav neopakuji.
        if (j>1)
        {
          for (k=(j-1); k>0; k--)
          {
            if (FA.getElementsByTagName('state')[i-1].get
      ElementsByTagName('transition')[k-1].getAttribute
      ('symbol') == FA.getElementsByTagName('state')[i-1].get
      ElementsByTagName('transition')[j-1].getAttribute
      ('symbol'))
              {
                alert("Znaky \"" +j+ "\" a \"" +k+ "\" maji pro
      stav \""+stav[i]+"\" stejny nazev!\n\""+znak[j]+"\"");
                chyba++;
              }
          }
        }
      }
    }
  }
// Kontrola1: pocatecni stav je jednim ze stavu v prechodove
      tabulce.
  var jeprvkem = false, chyba1 = true;
  while (chyba1)
  {
    for (i=0; i<q; i++)
    {
      if (q0 == stav[i])
        jeprvkem = true;
    }
    if (jeprvkem == false)
    {
      alert("Pocatecni stav neni z mnoziny stavu!");
      chyba1 = confirm("Zadat pocatecni stav znovu?");
      if (chyba1)
      {
        q0 = prompt("Zadej nazev pocatecniho stavu:", "");
      }
    }
    else chyba1 = false;
  }
```

```
// Kontrola3: v poli tabulky jsou stejne stavy jako v zahlavi
      tabulky, nebo je pole nevyplneno.
  var k, pole, jinystav;
  var chyba3 = true;
  while (chyba3)
  {
    pole = 0;
    for (i=1; i<q; i++)
    {
      for (j=1; j<s; j++)
      {
        jinystav = 0;
        for (k=1; k<q; k++)
        {
          if (delta[i][j] != stav[k])
            jinystav ++;
        }
        if (jinystav == (q-1) && delta[i][j] != "")
        {
          alert("Na radku \"" +i+ "\" a sloupci \"" +j+ "\" je
      stav \"" +delta[i][j]+ "\", ktery neni uveden v zahlavi
      tabulky!");
          delta[i][j] = prompt("Zmena stavu \"" +stav[i]+ "\"
      po znaku \"" +znak[j]+ "\": ", "");
          pole++;
        }
      }
    }
    if (pole == 0)
    chyba3 = false;
  }

// Kontrola4: v zahlavi tabulky nejsou nedosazitelne stavy;
      neprovadi se pro pocatecni stav
  for (k=1; k<q; k++)
  {
    var mimo = 0;
    if (stav[k] != q0)
    {
      for (i=1; i<q; i++)
      {
        for (j=1; j<q; j++)
        {
           if (k != i && stav[k] == delta[i][j])
           mimo ++;
         }
      }
      if (mimo = 0)
        alert("Stav \"" +stav[k]+ "\" je nedosazitelny!");
    }
  }
```

```
// Kontrola5: existuje alespon jeden rozpoznavaci stav.
  var F = 0;
    for (i=1; i<q; i++)
    {
      if (konec[i] == "true")
        F++;
    }
    if (F == 0)
    {
      alert("Nebyl nalezen zadny rozpoznavaci stav!");
      for (i=1; i<q; i++)
      {
        konec[i] = prompt("Je stav \"" +stav[i]+ "\"
      rozpoznavaci?\ntrue/false", "true");
      }
    }
    else
      chyba5 = false;
// Kontrola6: upozorneni na opakovani radku pro
      (ne)rozpoznavaci stavy
  for (i=1; i<q-1; i++)
  {
    var stejne = 0;
    for (j=i+1; j<q; j++)
    {
      if (i != j && konec[i] == konec[j])
      {
        for (k=1; k<s; k++)
        {
          if (i != j && delta[i][k] == delta[j][k])
            stejne++;
        }
      }
      if (stejne >= (q-1))
      {
        alert("Stavy \"" +stav[i]+ "\" a \"" +stav[j]+ "\" jsou
      ekvivalentni!");
        chyba++;
      }
    }
  }
  if (chyba>0)
  {
    alert('Opravte chyby v souboru a načtěte jej znovu!')
  }
  else
  {
    var y = document.getElementById('popis_tab');
    y.removeAttribute('hidden', 'true');
    var tab = document.getElementById('tabKA');
    while (tab.hasChildNodes())
    {
      tab.removeChild(tab.lastChild);
    }
```

```
// vypis nazvu, mnoziny stavu, znaku a pocatecniho stavu
    var div = document.getElementById('popis');
    while (div.hasChildNodes())
    {
      div.removeChild(div.lastChild);
    }
    var nazevKA = document.createElement('p')
    nazevKA.textContent = "Název KA: "+nazev;
    div.appendChild(nazevKA);

    var mnozinaQ="Množina stavů: ";
    for (i=1; i<q; i++)
    {
      mnozinaQ += stav[i]+" ";
    }
    var stavyKA = document.createElement('p')
    stavyKA.textContent = mnozinaQ;
    div.appendChild(stavyKA);

    var mnozinaS="Vstupní abeceda: ";
    for (i=1; i<s; i++)
    {
      mnozinaS += znak[i]+" ";
    }
    var abecedaKA = document.createElement('p')
    abecedaKA.textContent = mnozinaS;
    div.appendChild(abecedaKA);

    var psKA = document.createElement('p')
    psKA.textContent = "Počáteční stav: "+q0;
    div.appendChild(psKA);

    var mnozinaF="Rozpoznávací stavy: ";
    for (i=1; i<q; i++)
    {
      if (konec[i]=="true")
        mnozinaF += stav[i]+" ";
    }
    var rozpozKA = document.createElement('p')
    rozpozKA.textContent = mnozinaF;
    div.appendChild(rozpozKA);

    var deltaKA = document.createElement('p')
    deltaKA.textContent = "Přechodová funkce zadaná tabulkou:";
    div.appendChild(deltaKA);

// tabulka prechodove funkce
    for (i=0; i<q; i++)
    {
      var newRow = document.createElement('TR');
      tab.appendChild(newRow);
      for (j=0; j<s; j++)
      {
        var newCell = document.createElement('TD');
```

```
        newCell.setAttribute('align', 'center');
        newRow.appendChild(newCell);
        if (i == 0 && j>0)
          newCell.textContent=znak[j];
        if (i != 0 && j==0)
          newCell.textContent=stav[i];
        if (i != 0 && j>0)
          newCell.textContent=delta[i][j];
      }
    }
  }

  var an = document.getElementById('KA_animace');
  an.removeAttribute('hidden', 'true');
  var div = document.getElementById('schema');
  while (div.hasChildNodes())
  {
    div.removeChild(div.lastChild);
    alert(div.hasChildNodes());
  }

  var u;

  var svg =
      document.createElementNS('http://www.w3.org/2000/svg',
      'svg');
  svg.setAttribute('width', Math.round(q/2)*80+250);
  svg.setAttribute('height', Math.round(q/2)*80+300);
  svg.setAttribute('id', 'svg');
  div.appendChild(svg);

// urceni souradnic stredu pro stavy - kruhy
  var x = new Array();
  var y = new Array();
  var c, delka, mocnina, alfa;
  delka = 200;
  x[1] = 200+Math.round(q/4)*50;
  y[1] = Math.round(q/2)*50+200;
  x[2] = x[1]-delka;
  y[2] = y[1];
  for (i=1; i<q; i++)
  {
    mocnina = Math.pow(-1,i);
    if (i>2)
    {
      alfa = 3.14*(1-2/(q-1))*(i-2);
      x[i] = Math.round(x[i-1]-mocnina*delka*Math.cos(alfa));
      y[i] = Math.round(y[i-1]+mocnina*delka*Math.sin(alfa));
    }
```

```
// oznaceni rozpoznavaciho stavu dvojitou kruznici + priprava
      pro animaci
    if (konec[i]=="true")
    {
      var circle_k =
      document.createElementNS('http://www.w3.org/2000/svg',
      'circle');
      circle_k.setAttribute('id', 'rskruh'+i);
      circle_k.setAttribute('cx', x[i]);
      circle_k.setAttribute('cy', y[i]);
      circle_k.setAttribute('r', '43');
      circle_k.setAttribute('style', 'stroke: black; fill:
      none; stroke-width: 2');
      svg.appendChild(circle_k);
    }

// oznaceni pocatecniho stavu sipkou ke kruznici
    if (stav[i] == q0)
    {
      var poc =
      document.createElementNS('http://www.w3.org/2000/svg',
      'line');
      poc.setAttribute('style', 'stroke: black; stroke-width:
      2');
      poc.setAttribute('x1', x[i]-100);
      poc.setAttribute('y1', y[i]);
      poc.setAttribute('x2', x[i]-55);
      poc.setAttribute('y2', y[i]);
      poc.setAttribute('marker-end', 'url(#sipka)');
      svg.appendChild(poc);
    }

// kruh + priprava pro jeho blikani a popisek pro znazorneni
      stavu
    var circle_s =
      document.createElementNS('http://www.w3.org/2000/svg',
      'circle');
    circle_s.setAttribute('id', stav[i]);
    circle_s.setAttribute('cx', x[i]);
    circle_s.setAttribute('cy', y[i]);
    circle_s.setAttribute('r', '40');
    circle_s.setAttribute('style', 'stroke: black; fill: none;
      stroke-width: 2');
    if (stav[i] == q0)
    {
      circle_s.setAttribute('style', 'stroke: black; fill:
      lightgray; stroke-width: 2');
    }
    svg.appendChild(circle_s);

    var text_stav =
      document.createElementNS('http://www.w3.org/2000/svg',
      'text');
    text_stav.setAttribute('id', 'text'+i);
```

```
    text_stav.setAttribute('x', x[i]-20);
    text_stav.setAttribute('y', y[i]+5);
    text_stav.textContent=stav[i];
    svg.appendChild(text_stav);
  }

  for (i=1; i<q; i++)
  {
    for (j=1; j<s; j++)
    {
// oblouk a popisek pro zobrazeni stavu na sebe samého
      if (stav[i] == delta[i][j])
      {
        var ox, oy;
        ox = x[i]-10;
        oy = y[i]-50;
        var oblouk =
      document.createElementNS('http://www.w3.org/2000/svg',
      'path');
        oblouk.setAttribute('d','M '+ox+' '+oy+' a 30 30 0 1 1
      20 0');
        oblouk.setAttribute('marker-end', 'url(#sipka)');
        oblouk.setAttribute('style', 'stroke: black; fill:
      none; stroke-width: 2');
        svg.appendChild(oblouk);

        ox = x[i]-25;
        oy = y[i]-75;
        var text_oblouk =
      document.createElementNS('http://www.w3.org/2000/svg',
      'text');
        text_oblouk.setAttribute('x', ox);
        text_oblouk.setAttribute('y', oy);
        text_oblouk.textContent=znak[j];
        svg.appendChild(text_oblouk);
      }

// sipka pro ostatni prechody
      else
      {
        for (k=1; k<q; k++)
        {
          if (i!=k && delta[i][j]==stav[k])
          {
            var beta, gama, xs, ys, xt, yt, xp, yp;
            beta = Math.round(Math.atan
      ((y[k]-y[i])/(x[k]-x[i])));
            gama = Math.atan((x[k]-x[i])/(y[k]-y[i]));
            if (x[k]>x[i])
            {
              if (y[k]>y[i])
              {
                xs = x[i]+Math.round(40*Math.cos(beta))+3;
                xt = x[k]-Math.round(45*Math.cos(beta))+3;
```

```
      ys = y[i]+Math.round(40*Math.sin(beta))-3;
      yt = y[k]-Math.round(45*Math.sin(beta))-3;
      xp = x[i]+(x[k]-x[i])/2-12;
      yp = y[i]+(y[k]-y[i])/2+12;
    }
    else if (y[k]==y[i])
    {
      xs = x[i]+40;
      xt = x[k]-45;
      ys = y[i]-5;
      yt = ys;
      xp = x[i]+(x[k]-x[i])/2;
      yp = y[i]-15;
    }
    else
    {
      xs = x[i]+Math.round(40*Math.cos(beta))+3;
      xt = x[k]-Math.round(45*Math.cos(beta))+3;
      ys = y[i]+Math.round(40*Math.sin(beta))+3;
      yt = y[k]-Math.round(45*Math.sin(beta))+3;
      xp = x[i]+(x[k]-x[i])/2+5;
      yp = y[i]+(y[k]-y[i])/2+15;
    }
  }
  else if (x[k]<x[i])
    {
      if (y[k]>y[i])
      {
        xs = x[i]+Math.round(40*Math.sin(gama))-3;
        xt = x[k]-Math.round(45*Math.sin(gama))-3;
        ys = y[i]+Math.round(40*Math.cos(gama))-3;
        yt = y[k]-Math.round(45*Math.cos(gama))-3;
        xp = x[i]+(x[k]-x[i])/2-20;
        yp = y[i]+(y[k]-y[i])/2-10;
      }
      else if (y[k]==y[i])
      {
        xs = x[i]-40;
        xt = x[k]+45;
        ys = y[i]+5;
        yt = ys;
        xp = x[i]+(x[k]-x[i])/2;
        yp = y[i]+20;
      }
      else
      {
        xs = x[i]-Math.round(40*Math.sin(gama))-3;
        xt = x[k]+Math.round(45*Math.sin(gama))-3;
        ys = y[i]-Math.round(40*Math.cos(gama))+3;
        yt = y[k]+Math.round(45*Math.cos(gama))+3;
        xp = x[i]+(x[k]-x[i])/2+5;
        yp = y[i]+(y[k]-y[i])/2-10;
      }
    }
```

```
                else
                {
                  if(y[k]<y[i])
                  {
                    xs = x[i]-3;
                    xt = xs;
                    ys = y[i]-40;
                    yt = y[k]+45;
                    xp = x[i]-15;
                    yp = y[i]+(y[k]-y[i])/2;
                  }
                  else
                  {
                    xs = x[i]+3;
                    xt = xs;
                    ys = y[i]+40;
                    yt = y[k]-45;
                    xp = x[i]+10;
                    yp = y[i]+(y[k]-y[i])/2-5;
                  }
                }

                var prechod =
      document.createElementNS('http://www.w3.org/2000/svg',
      'line');
                prechod.setAttribute('id', 'stav'+i+k);
                prechod.setAttribute('style', 'stroke: black;
      stroke-width: 2');
                prechod.setAttribute('x1', xs);
                prechod.setAttribute('y1', ys);
                prechod.setAttribute('x2', xt);
                prechod.setAttribute('y2', yt);
                prechod.setAttribute('marker-end', 'url(#sipka)');
                svg.appendChild(prechod);

                var text_prechod =
      document.createElementNS('http://www.w3.org/2000/svg',
      'text');
                text_prechod.setAttribute('x', xp);
                text_prechod.setAttribute('y', yp);
                text_prechod.textContent=znak[j];
                svg.appendChild(text_prechod);
              }
            }
          }
        }
      }

// vytvoreni symbolu sipky na konec oblouku a usecek
      var sipka =
          document.createElementNS('http://www.w3.org/2000/svg',
          'defs');
```

```
  var marker =
      document.createElementNS('http://www.w3.org/2000/svg',
      'marker');
  marker.setAttribute('id', 'sipka');
  marker.setAttribute('viewBox', '0 0 10 10');
  marker.setAttribute('refX', '0');
  marker.setAttribute('refY', '5');
  marker.setAttribute('markerUnits', 'strokeWidth');
  marker.setAttribute('markerWidth', '4');
  marker.setAttribute('markerHeight', '3');
  marker.setAttribute('orient', 'auto');
  sipka.appendChild(marker);
  svg.appendChild(sipka);

  var k_sipce =
      document.createElementNS('http://www.w3.org/2000/svg',
      'path');
  k_sipce.setAttribute('d', 'M 0 0 L 10 5 L 0 10 z');
  marker.appendChild(k_sipce);
}
```

6.3.2. DIALOG.JS

```
function dialog()
{
  var vypni=document.getElementById('sd');
  vypni.setAttribute('hidden', 'true');
  var ukaz=document.getElementById('z_udaje');
  ukaz.removeAttribute('hidden', 'true');
  var vypln = document.getElementById('vyplneni');
  vypln.setAttribute('hidden', 'true');

  var tab=document.getElementById('dialog_tab1');
//  když bude mít tabulka děti, tak se odstrani
  while (tab.hasChildNodes())
  {
    tab.removeChild(tab.lastChild);
  }
    var radek1 = document.createElement('TR');
    tab.appendChild(radek1);
    var bunka1 = document.createElement('TD');
    radek1.appendChild(bunka1);
    var popisek1 = document.createTextNode('Název KA');
    bunka1.appendChild(popisek1);
    var bunka2 = document.createElement('TD');
    radek1.appendChild(bunka2);
    var it = document.createElement('input');
    it.setAttribute("type", "text");
    it.setAttribute("id", "nazev");
    it.setAttribute("value", "Název KA") ;
    bunka2.appendChild(it);
```

```
    var radek2 = document.createElement('TR');
    tab.appendChild(radek2);
    var bunka12 = document.createElement('TD');
    radek2.appendChild(bunka12);
    var newText2 = document.createTextNode('Počet stavů');
    bunka12.appendChild(newText2);
    var bunka22 = document.createElement('TD');
    radek2.appendChild(bunka22);
    var newInput2 = document.createElement('input');
    newInput2.setAttribute("type", "text");
    newInput2.setAttribute("id", "q");
    newInput2.setAttribute("value", "Počet stavů") ;
    bunka22.appendChild(newInput2);

    var radek3 = document.createElement('TR');
    tab.appendChild(radek3);
    var bunka13 = document.createElement('TD');
    radek3.appendChild(bunka13);
    var newText3 = document.createTextNode('Počet znaků');
    bunka13.appendChild(newText3);
    var bunka23 = document.createElement('TD');
    radek3.appendChild(bunka23);
    var newInput3 = document.createElement('input');
    newInput3.setAttribute("type", "text");
    newInput3.setAttribute("id", "s");
    newInput3.setAttribute("value", "Počet znaků") ;
    bunka23.appendChild(newInput3);

//    <input type="button" value="nacti data"
      onClick="kontrola1()">
    var radek4 = document.createElement('TR');
    radek4.setAttribute("id", "tlacitka");
    tab.appendChild(radek4);
    var bunka41 = document.createElement('TD');
    radek4.appendChild(bunka41);
    var tl = document.createElement('input');
    tl.setAttribute('id', 'tl');
    tl.setAttribute("type", "button");
    tl.setAttribute("value", "Načíst data");
    tl.setAttribute("onClick", "kontrola1()");
    bunka41.appendChild(tl);
}

function kontrola1()
{
  var qtab, stab, nazev, q, s;
  qtab=document.getElementById("q");
  stab=document.getElementById("s");

  q = parseInt(qtab.value);
  s = parseInt(stab.value);

  var chyba = 0;
```

```
  var cislo = /[1-9][0-9]?/;

  if (!cislo.test(q))
  {
    alert("Počet stavů musí být přirozené číslo!");
    chyba++;
  }
  if (!cislo.test(s))
  {
    alert("Počet stavů musí být přirozené číslo!");
    chyba++;
  }
  if (chyba==0)
  {
//  <input type="button" value="vytvor deltu"
      onClick="delta()">
    delta_zadani();
  }
}

function delta_zadani()
{
  var z = document.getElementById('z_udaje');
  z.setAttribute('hidden', 'true');
  var y = document.getElementById('vyplneni');
  y.removeAttribute('hidden', 'true');
  var nazevtab = document.getElementById("nazev");
  var nazev = nazevtab.value;
  qtab = document.getElementById("q");
  q = parseInt(qtab.value)+1;
  stab = document.getElementById("s");
  s = parseInt(stab.value)+3;

  var tab=document.getElementById('dialog_tab2');

  while (tab.hasChildNodes())
  {
    tab.removeChild(tab.lastChild);
  }
  for (i=0; i<q; i++)
  {
    var newRow = document.createElement('TR');
    tab.appendChild(newRow);
    for (j=0; j<s; j++)
    {
      var newCell = document.createElement('TD');
      newCell.setAttribute('align', 'center');
      newRow.appendChild(newCell);

      if (i == 0 && j==0)
      {
        var text = document.createTextNode("Počáteční stav");
        newCell.appendChild(text);
      }
```

```
      if (i == 0 && j==1)
      {
        var text = document.createTextNode("Rozpoznávací
        stavy");
        newCell.appendChild(text);
      }
      if (i == 0 && j==2)
      {
        var text = document.createTextNode(nazev);
        newCell.appendChild(text);
      }
      if (i == 0 && j>2)
      {
        var it = document.createElement('input');
        it.setAttribute("type", "text");
        it.setAttribute("id", "delta"+i+(j-2));
        it.setAttribute("value", "znak"+(j-2));
        newCell.appendChild(it);
      }
      if (i != 0 && j==0)
      {
        var ir = document.createElement('input');
        ir.setAttribute("type", "radio");
        ir.setAttribute("name", "pocatek");
        ir.setAttribute("id", "start"+i);
        newCell.appendChild(ir);
        if (i == 1)
        ir.setAttribute("checked", "checked");
      }
      if (i != 0 && j==1)
      {
        var ic = document.createElement('input');
        ic.setAttribute("type", "checkbox");
        ic.setAttribute("id", "konec"+i);
        newCell.appendChild(ic);
      }
      if (i != 0 && j==2)
      {
        var it = document.createElement('input');
        it.setAttribute("type", "text");
        it.setAttribute("id", "delta"+i+(j-2));
        it.setAttribute("value", "stav"+i);
        newCell.appendChild(it);
      }
      if (i != 0 && j>2)
      {
        var it = document.createElement('input');
        it.setAttribute("type", "text");
        it.setAttribute("id", "delta"+i+(j-2));
        it.setAttribute("value", "stav"+i+(j-2));
        newCell.appendChild(it);
      }
    }
  }
```

```
//      <input type="button" value="zkontroluj"
        onClick="kontrola2()">
    var newRow = document.createElement('TR');
    tab.appendChild(newRow);

    var newCell = document.createElement('TD');
    newRow.appendChild(newCell);
    var tl= document.createElement('input');
    tl.setAttribute('type', 'button');
    tl.setAttribute('value', 'Zkontrolovat');
    tl.setAttribute('onclick', 'kontrola2()');
    newCell.appendChild(tl);

    var newCell_oprava = document.createElement('TD');
    newRow.appendChild(newCell_oprava);
    var tl2= document.createElement('input');
    tl2.setAttribute('type', 'button');
    tl2.setAttribute('value', 'Opravit první tabulku');
    tl2.setAttribute('onclick', 'dialog()');
    newCell_oprava.appendChild(tl2);
}

function kontrola2()
{
  var i, j, q, s, delta, konec, q0, nazev, nazevtab, qtab,
        stab, deltatab, znak, stav, chyba;
  chyba=0;
//nacteni nazvu KA, poctu stavu a znaku
  nazevtab = document.getElementById("nazev");
  nazev = nazevtab.value;
  qtab = document.getElementById("q");
  q = parseInt(qtab.value)+1;
  stab = document.getElementById("s");
  s = parseInt(stab.value)+1;
//nacteni stavu
  stav = new Array();
  for (i=1; i<q; i++)
  {
    deltatab = document.getElementById("delta"+i+"0");
    stav[i] = deltatab.value;
//nazvy stavu se neopakuji
    if (i>1)
    {
      for (j=(i-1); j>0; j--)
      {
        if (stav[i] == stav[j])
        {
          alert("Stavy "+i+" a "+j+ " mají stejný název:
        "+stav[i]+"!");
          chyba++;
        }
      }
    }
  }
```

```
//znak je jedno pismeno nebo cislice
  var symbol = /^(\w){1}$/;

//nacteni znaku
  znak = new Array();
  for (i=1; i<s; i++)
  {
    deltatab = document.getElementById("delta0"+i);
    znak[i] = deltatab.value;
// znak je jedno písmeno, podtržítko nebo číslice
    if (!symbol.test(znak[i]))
    {
      alert("Znak"+i+" \""+znak[i]+"\" není jedno písmeno nebo
        jedna číslice!");
      chyba++;
    }
// nazvy znaku se neopakuji
    if (i>1)
    {
      for (j=(i-1); j>0; j--)
      {
        if (znak[i] == znak[j])
        {
          alert("Znaky "+i+" a "+j+ " mají stejný název:
        "+znak[i]+"!");
          chyba++;
        }
      }
    }
  }

//pocatecni stav
  for (i=1; i<q; i++)
  {
    qtab = document.getElementById("start"+i);
    if (qtab.checked)
      q0 = stav[i];
  }

//rozpoznavaci stavy
  var rs=0;
  konec = new Array();

  for (i=1; i<q; i++)
  {
    konec[i] = false;
    qtab = document.getElementById("konec"+i);
    if (qtab.checked)
    {
      konec[i] = "true";
      rs++;
    }
  }
```

```
   if (rs==0)
   {
    alert("Musí být označen alespoň jeden rozpoznávací stav!");
    chyba++;
   }


//nacteni prechodu
  var k, n, m;
  delta = new Array();
  for (i=1; i<q; i++)
  {
    m=0;
    delta[i] = new Array();
    for (j=1; j<s; j++)
    {
      n=0;
      deltatab = document.getElementById("delta"+i+j);
      delta[i][j] = deltatab.value;
//stavy po prechodu odpoviadaji stavum v zahlavi
      if (delta[i][j] == "null")
      {
        alert("Přechod stavu "+i+" po znaku "+j+" není
        zadán!");
        chyba++;
      }
      else
      {
        for (k=1; k<q; k++)
        {
          if (stav[k] != delta[i][j])
            n++;
        }
        if (n == (q-1))
        {
          alert("Stav "+delta[i][j]+" jako přechod stavu
          "+i+"po znaku "+j+" není z množiny stavů!");
          chyba++;
        }
      }
//opakovani radku
      if (i>1)
      {
        for (k=(i-1); k>0;k--)
        {
          if (delta[i][j] == delta[k][j] && konec[i] ==
          konec[k])
          {
            m++;
            l=k;
          }
        }
      }
    }
```

```
    if (m==(s-1))
    {
      alert("Řádky "+i+" a " +l+ " jsou shodné!");
      chyba++;
    }
  }

//upozorneni na pripadne nedosazitelne stavy, neprovadi se pro
        pocatecni stav
    for (k=1; k<q; k++)
    {
      var mimo = 0;
      if (stav[k] != q0)
      {
        for (i=1; i<q; i++)
        {
          for (j=1; j<q; j++)
          {
            if (k != i && stav[k] == delta[i][j])
            {
              mimo ++;
            }
          }
        }
        if (mimo == 0)
        {
          alert("Stav \"" +stav[k]+ "\" je nedosažitelný!");
          chyba++;
        }
      }
    }
  if (chyba>0)
  {
    alert("Po opravě chyb znovu stiskněte tlačítko
        \"Zkontrolovat\" !");
  }
  else
  {
    alert("Vše je OK");
    popisKA();
    schemaKA();
    var vypni=document.getElementById('sd');
    vypni.removeAttribute('hidden', 'true');
//vymazat všechny potomky tabulky dialog_tab1
// var tab=document.getElementById('dialog_tab1');
  }
}
```

6.3.3. POPISKA.JS

```
function popisKA()
{
  var z = document.getElementById('vyplneni');
  z.setAttribute('hidden', 'true');
  var y = document.getElementById('popis_tab');
  y.removeAttribute('hidden', 'true');

  var tab = document.getElementById('tabKA');
  while (tab.hasChildNodes())
  {
    tab.removeChild(tab.lastChild);
  }

// nacteni nazvu, poctu stavu a znaku
  var nazevtab = document.getElementById("nazev");
  var nazev = nazevtab.value;
  qtab = document.getElementById("q");
  q = parseInt(qtab.value)+1;
  stab = document.getElementById("s");
  s = parseInt(stab.value)+1;
// vypis nazvu, mnoziny stavu, znaku a pocatecniho stavu
  var div = document.getElementById('popis');
  while (div.hasChildNodes())
  {
    div.removeChild(div.lastChild);
  }

  var nazevKA = document.createElement('p')
  nazevKA.textContent = "Název KA: "+nazev;
  div.appendChild(nazevKA);

  var mnozinaQ="Množina stavů: ";
  for (i=1; i<q; i++)
  {
    var z = document.getElementById('delta'+i+'0');
    mnozinaQ += z.value+" ";
  }
  var stavyKA = document.createElement('p')
  stavyKA.textContent = mnozinaQ;
  div.appendChild(stavyKA);

  var mnozinaS="Vstupní abeceda: ";
  for (i=1; i<s; i++)
  {
    var z = document.getElementById('delta0'+i);
    mnozinaS += z.value+" ";
  }
  var abecedaKA = document.createElement('p')
  abecedaKA.textContent = mnozinaS;
  div.appendChild(abecedaKA);
```

```
  for (i=1; i<q; i++)
  {
    var z = document.getElementById('start'+i);
    var y = document.getElementById('delta'+i+"0");
    if (z.checked)
    {
      var psKA = document.createElement('p')
      psKA.textContent = "Počáteční stav: "+y.value;
      div.appendChild(psKA);
    }
  }

  var mnozinaF="Rozpoznávací stavy: ";
  for (i=1; i<q; i++)
  {
    var z = document.getElementById("konec"+i);
    var y = document.getElementById('delta'+i+"0");
    if (z.checked)
      mnozinaF += y.value+" ";
  }
  var rozpozKA = document.createElement('p')
  rozpozKA.textContent = mnozinaF;
  div.appendChild(rozpozKA);

  var deltaKA = document.createElement('p')
  deltaKA.textContent = "Přechodová funkce zadaná tabulkou:";
  div.appendChild(deltaKA);

// tabulka prechodove funkce
  for (i=0; i<q; i++)
  {
    var newRow = document.createElement('TR');
    tab.appendChild(newRow);
    for (j=0; j<s; j++)
    {
      var newCell = document.createElement('TD');
      newCell.setAttribute('align', 'center');
      newRow.appendChild(newCell);

      if (i == 0 && j>0)
      {
        var z = document.getElementById('delta'+i+j);
        var it = document.createTextNode(z.value);
        newCell.appendChild(it);
      }

      if (i != 0 && j==0)
      {
        var z = document.getElementById('delta'+i+j);
        var it = document.createTextNode(z.value);
        newCell.appendChild(it);
      }
```

```
      if (i != 0 && j>0)
      {
        var z = document.getElementById('delta'+i+j);
        var it = document.createTextNode(z.value);
        newCell.appendChild(it);
      }
    }
  }
}
```

6.3.4. SCHEMAKA.JS

```
function schemaKA()
{
  var nazevtab = document.getElementById("nazev");
  var nazev = nazevtab.value;
  qtab = document.getElementById("q");
  q = parseInt(qtab.value)+1;
  stab = document.getElementById("s");
  s = parseInt(stab.value)+1;
  var q0, i, j, z;
  var stav = new Array();
  var znak = new Array();
  var delta = new Array();
  var konec = new Array();
  for (i=1; i<q; i++)
  {
    delta[i] = new Array();
// nacteni stavu v zahlavi
    z = document.getElementById('delta'+i+'0');
    stav[i] = z.value;
    for (j=1; j<s; j++)
    {
// nacteni znaku
      z = document.getElementById('delta0'+j);
      znak[j] = z.value
//nacteni noveho stavu po prechodu
      z = document.getElementById('delta'+i+j);
      delta[i][j] = z.value;
    }
//nacteni rozpoznavacich stavu
    z = document.getElementById('konec'+i);
    if (z.checked)
      konec[i] = "true";
    else
      konec[i] = "false";
//nacteni pocatecniho stavu
    z = document.getElementById('start'+i);
    if (z.checked)
      q0 = stav[i];
  }
```

```
  var an = document.getElementById('KA_animace');
  an.removeAttribute('hidden', 'true');
  var div = document.getElementById('schema');
  while (div.hasChildNodes())
  {
    div.removeChild(div.lastChild);
  }
  var u;
  var svg =
        document.createElementNS('http://www.w3.org/2000/svg',
        'svg');
  svg.setAttribute('width', Math.round(q/2)*120+150);
  svg.setAttribute('height', Math.round(q/2)*130+50);
  svg.setAttribute('id', 'svg');
  div.appendChild(svg);
// urceni souradnic stredu pro stavy - kruhy
  var x = new Array();
  var y = new Array();
  var c, delka, mocnina, alfa;
  delka = 200;
  if (q<7)
    x[1] = Math.round(q/2)*130;
  else if (q<11)
    x[1] = Math.round(q/2)*100;
  else
    x[1] = Math.round(q/2)*80;
  y[1] = Math.round(q/2)*130;
  x[2] = x[1]-delka;
  y[2] = y[1];
  for (i=1; i<q; i++)
  {
    mocnina = Math.pow(-1,i);
    if (i>2)
    {
      alfa = 3.14*(1-2/(q-1))*(i-2);
      x[i] = Math.round(x[i-1]-mocnina*delka*Math.cos(alfa));
      y[i] = Math.round(y[i-1]+mocnina*delka*Math.sin(alfa));
    }
// oznaceni rozpoznavaciho stavu dvojitou kruznici + priprava
        pro animaci
    if (konec[i]=="true")
    {
      var circle_k =
        document.createElementNS('http://www.w3.org/2000/svg',
        'circle');
      circle_k.setAttribute('id', 'rskruh'+i);
      circle_k.setAttribute('cx', x[i]);
      circle_k.setAttribute('cy', y[i]);
      circle_k.setAttribute('r', '43');
      circle_k.setAttribute('style', 'stroke: black; fill:
        none; stroke-width: 2');
      svg.appendChild(circle_k);
    }
```

```
// oznaceni pocatecniho stavu sipkou ke kruznici
    if (stav[i] == q0)
    {
      var poc =
        document.createElementNS('http://www.w3.org/2000/svg',
        'line');
      poc.setAttribute('style', 'stroke: black; stroke-
        width: 2');
      poc.setAttribute('x1', x[i]-100);
      poc.setAttribute('y1', y[i]);
      poc.setAttribute('x2', x[i]-55);
      poc.setAttribute('y2', y[i]);
      poc.setAttribute('marker-end', 'url(#sipka)');
      svg.appendChild(poc);
    }
// kruh + priprava pro jeho blikani a popisek pro znazorneni
        stavu
    var circle_s =
        document.createElementNS('http://www.w3.org/2000/svg',
        'circle');
    circle_s.setAttribute('id', stav[i]);
    circle_s.setAttribute('cx', x[i]);
    circle_s.setAttribute('cy', y[i]);
    circle_s.setAttribute('r', '40');
    circle_s.setAttribute('style', 'stroke: black; fill: none;
        stroke-width: 2');
    if (stav[i] == q0)
    {
      circle_s.setAttribute('style', 'stroke: black; fill:
        lightgray; stroke-width: 2');
    }
    svg.appendChild(circle_s);
    var text_stav =
        document.createElementNS('http://www.w3.org/2000/svg',
        'text');
    text_stav.setAttribute('id', 'text'+i);
    text_stav.setAttribute('x', x[i]-20);
    text_stav.setAttribute('y', y[i]+5);
    text_stav.textContent=stav[i];
    svg.appendChild(text_stav);
  }
  for (i=1; i<q; i++)
  {
    for (j=1; j<s; j++)
    {
// oblouk a popisek pro zobrazeni stavu na sebe samého
      if (stav[i] == delta[i][j])
      {
        var ox, oy;
        ox = x[i]-10;
        oy = y[i]-50;
        var oblouk =
        document.createElementNS('http://www.w3.org/2000/svg',
        'path');
```

```
        oblouk.setAttribute('d','M '+ox+' '+oy+'
         a 30 30 0 1 1 20 0');
        oblouk.setAttribute('marker-end', 'url(#sipka)');
        oblouk.setAttribute('style', 'stroke: black; fill:
        none; stroke-width: 2');
        svg.appendChild(oblouk);
        ox = x[i]-25;
        oy = y[i]-75;
        var text_oblouk =
        document.createElementNS('http://www.w3.org/2000/svg',
        'text');
        text_oblouk.setAttribute('x', ox);
        text_oblouk.setAttribute('y', oy);
        text_oblouk.textContent=znak[j];
        svg.appendChild(text_oblouk);
      }
// sipka pro ostatni prechody
      else
      {
        for (k=1; k<q; k++)
        {
          if (i!=k && delta[i][j]==stav[k])
          {
            var beta, gama, xs, ys, xt, yt, xp, yp;
            beta = Math.atan((y[k]-y[i])/(x[k]-x[i]));
            gama = Math.atan((x[k]-x[i])/(y[k]-y[i]));
            if (x[k]>x[i])
            {
              if (y[k]>y[i])
              {
                xs = x[i]+Math.round(40*Math.cos(beta))+3;
                xt = x[k]-Math.round(45*Math.cos(beta))+3;
                ys = y[i]+Math.round(40*Math.sin(beta))-3;
                yt = y[k]-Math.round(45*Math.sin(beta))-3;
                xp = x[i]+(x[k]-x[i])/2-12;
                yp = y[i]+(y[k]-y[i])/2+12;
              }
              else if (y[k]==y[i])
              {
                xs = x[i]+40;
                xt = x[k]-45;
                ys = y[i]-5;
                yt = ys;
                xp = x[i]+(x[k]-x[i])/2;
                yp = y[i]-15;
              }
              else
              {
                xs = x[i]+Math.round(40*Math.cos(beta))+3;
                xt = x[k]-Math.round(45*Math.cos(beta))+3;
                ys = y[i]+Math.round(40*Math.sin(beta))+3;
                yt = y[k]-Math.round(45*Math.sin(beta))+3;
                xp = x[i]+(x[k]-x[i])/2+5;
                yp = y[i]+(y[k]-y[i])/2+15;
```

```
    }
  }
  else if (x[k]<x[i])
    {
      if (y[k]>y[i])
      {
        xs = x[i]+Math.round(40*Math.sin(gama))-3;
        xt = x[k]-Math.round(45*Math.sin(gama))-3;
        ys = y[i]+Math.round(40*Math.cos(gama))-3;
        yt = y[k]-Math.round(45*Math.cos(gama))-3;
        xp = x[i]+(x[k]-x[i])/2-20;
        yp = y[i]+(y[k]-y[i])/2-10;
      }
      else if (y[k]==y[i])
      {
        xs = x[i]-40;
        xt = x[k]+45;
        ys = y[i]+5;
        yt = ys;
        xp = x[i]+(x[k]-x[i])/2;
        yp = y[i]+20;
      }
      else
      {
        xs = x[i]-Math.round(40*Math.sin(gama))-3;
        xt = x[k]+Math.round(45*Math.sin(gama))-3;
        ys = y[i]-Math.round(40*Math.cos(gama))+3;
        yt = y[k]+Math.round(45*Math.cos(gama))+3;
        xp = x[i]+(x[k]-x[i])/2+5;
        yp = y[i]+(y[k]-y[i])/2-10;
      }
    }
    else
    {
      if(y[k]<y[i])
      {
        xs = x[i]-3;
        xt = xs;
        ys = y[i]-40;
        yt = y[k]+45;
        xp = x[i]-15;
        yp = y[i]+(y[k]-y[i])/2;
      }
      else
      {
        xs = x[i]+3;
        xt = xs;
        ys = y[i]+40;
        yt = y[k]-45;
        xp = x[i]+10;
        yp = y[i]+(y[k]-y[i])/2-5;
      }
    }
```

```
            var prechod =
        document.createElementNS('http://www.w3.org/2000/svg',
        'line');
            prechod.setAttribute('id', 'stav'+i+k);
            prechod.setAttribute('style', 'stroke: black;
        stroke-width: 2');
            prechod.setAttribute('x1', xs);
            prechod.setAttribute('y1', ys);
            prechod.setAttribute('x2', xt);
            prechod.setAttribute('y2', yt);
            prechod.setAttribute('marker-end', 'url(#sipka)');
            svg.appendChild(prechod);

            var text_prechod =
        document.createElementNS('http://www.w3.org/2000/svg',
        'text');
            text_prechod.setAttribute('x', xp);
            text_prechod.setAttribute('y', yp);
            text_prechod.textContent=znak[j];
            svg.appendChild(text_prechod);
          }
        }
      }
    }
  }
// vytvoreni symbolu sipky na konec oblouku a usecek
  var sipka =
        document.createElementNS('http://www.w3.org/2000/svg',
        'defs');
  var marker =
        document.createElementNS('http://www.w3.org/2000/svg',
        'marker');
  marker.setAttribute('id', 'sipka');
  marker.setAttribute('viewBox', '0 0 10 10');
  marker.setAttribute('refX', '0');
  marker.setAttribute('refY', '5');
  marker.setAttribute('markerUnits', 'strokeWidth');
  marker.setAttribute('markerWidth', '4');
  marker.setAttribute('markerHeight', '3');
  marker.setAttribute('orient', 'auto');
  sipka.appendChild(marker);
  svg.appendChild(sipka);
  var k_sipce =
        document.createElementNS('http://www.w3.org/2000/svg',
        'path');
  k_sipce.setAttribute('d', 'M 0 0 L 10 5 L 0 10 z');
  marker.appendChild(k_sipce);
}
```

6.3.5. ANIMACE.JS

```
function animace()
{
  if (nacteni == 1)
    krok_xml();
  else
    krok();
}

function krok()
{
  qtab = document.getElementById("q");
  q = parseInt(qtab.value)+1;
  stab = document.getElementById("s");
  s = parseInt(stab.value)+1;
  var q0, i, j, z;
  var stav = new Array();
  var znak = new Array();
  var delta = new Array();
  var konec = new Array();
  for (i=1; i<q; i++)
  {
    delta[i] = new Array();
// nacteni stavu v zahlavi
    z = document.getElementById('delta'+i+'0');
    stav[i] = z.value;
    for (j=1; j<s; j++)
    {
// nacteni znaku
      z = document.getElementById('delta0'+j);
      znak[j] = z.value
//nacteni noveho stavu po prechodu
      z = document.getElementById('delta'+i+j);
      delta[i][j] = z.value;
    }
//nacteni rozpoznavacich stavu
    z = document.getElementById('konec'+i);
    if (z.checked)
      konec[i] = "true";
    else
      konec[i] = "false";
//nacteni pocatecniho stavu
    z = document.getElementById('start'+i);
    if (z.checked)
      q0 = stav[i];
  }

  var znaky, i, zn;
  var slovo = document.getElementById('vstup');
  znaky = slovo.value;
//vypis znaku vedle stavoveho diagramu
  zn = new Array();
```

```
  var x = 10;
  for (i=0; i<znaky.length; i++)
  {
    zn[i] = znaky.charAt(i);
    var svg = document.getElementById('svg');
    var text =
        document.createElementNS('http://www.w3.org/2000/svg',
        'text');
    for (j=1; j<s; j++)
    if (zn[i] == znak[j])
    {
      text.setAttribute('id', 'text'+i);
      text.setAttribute('onclick', 'onZnakClick_krok()');
      text.setAttribute('x', x);
      text.setAttribute('y', '20');
      text.textContent=zn[i];
      svg.appendChild(text);
      x += 15;
    }
  }
}

// aktivace animaci
function onZnakClick_krok()
{
  qtab = document.getElementById("q");
  q = parseInt(qtab.value)+1;
  stab = document.getElementById("s");
  s = parseInt(stab.value)+1;
  var q0, i, j, z;
  var znak = new Array();
  var delta = new Array();
  for (i=1; i<q; i++)
  {
    delta[i] = new Array();
    for (j=1; j<s; j++)
    {
// nacteni znaku
      z = document.getElementById('delta0'+j);
      znak[j] = z.value
//nacteni noveho stavu po prechodu
      z = document.getElementById('delta'+i+j);
      delta[i][j] = z.value;
    }
  }
  var stav = new Array();
  var text = new Array();
  for (i=1; i<q; i++)
  {
    var st = document.getElementById('delta'+i+'0').value;
    stav[i] = document.getElementById(st);
    text[i] = document.getElementById('text'+i);
  }
  var slovo = document.getElementById('vstup');
```

```
  var znaky = slovo.value;
  var zn = new Array();
  for (l=0; l<znaky.length; l++)
  {
    zn[l] = znaky.charAt(l);
    for (j=1; j<s; j++)
    {
      if (zn[l] == znak[j])
      {
        for (i=1; i<q; i++)
        {
          if (stav[i].getAttribute('style') == 'stroke: black;
        fill: lightgray; stroke-width: 2')
          {
            for (k=1; k<q; k++)
            {
              if (delta[i][j] == text[k].textContent)
              {
                stav[i].setAttribute('style', 'stroke: black;
        fill: none; stroke-width: 2');
                stav[k].setAttribute('style', 'stroke: black;
        fill: lightgray; stroke-width: 2');
                alert('ze stavu
        \"'+stav[i].getAttribute('id')+'\" prejdu  po znaku
        \"'+zn[l]+'\" do stavu
        \"'+stav[k].getAttribute('id')+'\".');
                break;
              }
            }
            break;
          }
        }
        break;
      }
    }
  }
}
function krok_xml()
{
  var FA = xmlDoc.getElementsByTagName('FA')[0];
  var state = FA.getElementsByTagName('state');
  var q=state.length+1;
  var trans = state[0].getElementsByTagName('transition');
  var s=trans.length+1;
  var nazev = FA.getAttribute("name");
  var q0 = FA.getAttribute("startstate");

// Vyplneni tabulky prechodove funkce delta a oznaceni
        rozpoznavacich stavu.
  var delta = new Array();
  var konec = new Array();
  var znak = new Array();
  var stav = new Array();
```

```
  for (i=0; i<q; i++)
  {
    delta[i] = new Array();
    konec[i] = new Array();
    for (j=0; j<s; j++)
    {
      if(i==0 && j!=0)
      {
        znak[j] = trans[j-1].getAttribute("symbol");
      }
      else if (i!=0 && j==0)
      {
        stav[i] = state[i-1].getAttribute("id");
        konec[i] = state[i-1].getAttribute("accepting");
      }
      else if (i!=0 && j!=0 &&
        FA.getElementsByTagName('state')[i-1].get
        ElementsByTagName('transition')[j-1].get
        Attribute('target') != "null")
      {
        delta[i][j] = FA.getElementsByTagName('state')[i-1].get
        ElementsByTagName('transition')[j-1].get
        Attribute('target');
      }
    }
  }
  var znaky, i, zn;
  var slovo = document.getElementById('vstup');
  znaky = slovo.value;
//vypis znaku vedle stavoveho diagramu
  zn = new Array();
  var x = 10;

  for (i=0; i<znaky.length; i++)
  {
    zn[i] = znaky.charAt(i);
    var svg = document.getElementById('svg');
    var text =
        document.createElementNS('http://www.w3.org/2000/svg',
        'text');
    for (j=1; j<s; j++)
    if (zn[i] == znak[j])
    {
      text.setAttribute('id', 'text'+i);
      text.setAttribute('onclick', 'onZnakClick_krok_xml()');
      text.setAttribute('x', x);
      text.setAttribute('y', '20');
      text.textContent=zn[i];
      svg.appendChild(text);
      x += 15;
    }
  }
}
```

```
// aktivace animace diagramu z xml souboru
function onZnakClick_krok_xml()
{
  var FA = xmlDoc.getElementsByTagName('FA')[0];
  var state = FA.getElementsByTagName('state');
  var q=state.length+1;
  var trans = state[0].getElementsByTagName('transition');
  var s=trans.length+1;
  var nazev = FA.getAttribute("name");
  var q0 = FA.getAttribute("startstate");

// Vyplneni tabulky prechodove funkce delta a oznaceni
        rozpoznavacich stavu.
  var delta = new Array();
  var konec = new Array();
  var znak = new Array();
  var stav = new Array();

  for (i=0; i<q; i++)
  {
    delta[i] = new Array();
    konec[i] = new Array();
    for (j=0; j<s; j++)
    {
      if(i==0 && j!=0)
      {
        znak[j] = trans[j-1].getAttribute("symbol");
      }
      else if (i!=0 && j==0)
      {
        stav[i] = state[i-1].getAttribute("id");
        konec[i] = state[i-1].getAttribute("accepting");
      }
      else if (i!=0 && j!=0 &&
        FA.getElementsByTagName('state')[i-
        1].getElementsByTagName('transition')[j-
        1].getAttribute('target') != "null")
      {
        delta[i][j] = FA.getElementsByTagName('state')
        [i-1].getElementsByTagName('transition')
        [j-1].getAttribute('target');
      }
    }
  }
  var text = new Array();
  var st = new Array();
  for (i=1; i<q; i++)
  {
    st[i] = document.getElementById(stav[i]);
    text[i] = document.getElementById('text'+i);
  }
  var slovo = document.getElementById('vstup');
  var znaky = slovo.value;
  var zn = new Array();
```

```
  for (l=0; l<znaky.length; l++)
  {
    zn[l] = znaky.charAt(l);
    for (j=1; j<s; j++)
    {
      if (zn[l] == znak[j])
      {
        for (i=1; i<q; i++)
        {
          if (st[i].getAttribute('style') == 'stroke: black;
        fill: lightgray; stroke-width: 2')
          {
            for (k=1; k<q; k++)
            {
              if (delta[i][j] == text[k].textContent)
              {
                st[i].setAttribute('style', 'stroke: black;
        fill: none; stroke-width: 2');
                st[k].setAttribute('style', 'stroke: black;
        fill: lightgray; stroke-width: 2');
                alert('ze stavu
        \"'+st[i].getAttribute('id')+'\" prejdu  po znaku
        \"'+zn[l]+'\" do stavu
        \"'+st[k].getAttribute('id')+'\".');
                break;
              }
            }
            break;
          }
        }
        break;
      }
    }
  }
}
```

6.4. NAPOVEDA.HTML

```
<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Napoveda</title>
  </head>
  <body>
    <input type="button" id="s_s" value="Skrýt nápovědu"
        onclick="window.location.href='index.html'"/>
    <p>
      Pro <b>zobrazení ukázkového deterministického konečného
        automatu </b>klepněte buď na tlačítko <i>Zobrazit popis
        a schéma </i> nebo na tlačítko <i>Zobrazit xml </i>
        v části <i>Konečný automat rozpoznávající čísla
        dělitelná 4</i>.
    </p>
    <p>
      Pro <b> zadání </b> konečného deterministického automatu
        <b>ze souboru </b>je potřeba dodržet navržený
        dialekt:</br>
      &lt;element xmlns="http://relax.org/ns/structure/1.0"
        name="KA"&gt;  </br>
        &lt;attribute name="name"/&gt; </br>
        &lt;attribute name="startstate"/&gt; </br>
      &lt;oneOrMore&gt; </br>
        &lt;element name="state"&gt; </br>
          &lt;attribute name="id"/&gt; </br>
          &lt;attribute name="accepting"/&gt;</br>
        &lt;oneOrMore&gt; </br>
          &lt;element name="transition"&gt; </br>
            &lt;attribute name="symbol"/&gt; </br>
            &lt;attribute name="target"/&gt; </br>
          &lt;/element&gt; </br>
        &lt;/oneOrMore&gt; </br>
        &lt;/element&gt; </br>
      &lt;/oneOrMore&gt; </br>
      &lt;/element&gt; </br>
      a soubor uložit s koncovkou <i>.xml</i>. </br>
      V úvodní stránce pak soubor vyhledejte s pomocí tlačítka
        <i>Procházet…</i> a klepněte na tlačítko <i>Načíst xml
        soubor </i>.
    </p>
    <p>
      Pro <b>zadání s pomocí dialogu</b> klepněte na tlačítko
        <i>Spustit dialog</i> v části <i>Definice vlastního
        automatu</i>.
      Do připravené tabulky zadejte počet stavů, počet znaků
        vstupní abecedy a název Vašeho automatu. Klepněte na
        tlačítko <i>Načíst data</i>, provede se kontrola.<br/>
      Po správném zadání se zobrazí tabulka pro definici
        přechodové funkce. Až ji vyplníte, klepněte na tlačítko
```

```
        <i>Zkontrolovat</i>. Po správném zadání se zobrazí
        popis a stavový diagram Vámi zadaného deterministického
        konečného automatu.</br>
      Na první tabulku se ze druhé můžete vrátit stisknutím
        tlačítka <i>Opravit první tabulku</i>.
    </p>
    <p>
      <b>Popis konečného automatu</b> se generuje sám po
        správném zadání všech jeho parametrů.
    </p>
    <p>
      <b>Stavový diagram</b> se generuje automaticky po
        správném zadání všech parametrů automatu.
    </p>
    <p>
      Pro spuštění <b>animace</b> je potřeba nejprve zadat
        vstupní slovo a klepnout na tlačítko <i>Animovat</i>.
        Dále pak klepněte na jakýkoliv z vypsaných znaků -
        spustí se komentovaná animace, která s Vámi projde
        všechny přechody Vašeho konečného automatu.</br>
      Zadané znaky, které nepatří do vstupní abecedy jsou
        ignorovány.
    </p>
  </body>
</html>
```